Action History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes

Qun Shi

Abstract

This thesis presents a novel spatiotemporal action editing framework that synthesizes spatiotemporally synchronized multi-party interaction 3D video scenes from separately captured data.

Our main idea is to model multi-party interaction events with synchronization among body actions and gaze actions from multiple objects, and use them as constraints to perform spatiotemporal editing on 3D video data.

To achieve that, we first propose the idea of Action History Volume, which is a 3D volume data that encodes both spatial and temporal information of the object's action. This AHV based representation makes it possible to consider a duration of action as a whole in the editing work, instead of dealing with each single frames independently.

Then by defining the types of spatiotemporal synchronization between multiple AHVs, we build up a multi-party interaction dictionary that covers all types of interactions for our editing task. Taking advantage of the AHV based multi-party interaction dictionary, we enable to model multi-party interaction events into spatiotemporal constraints considering the synchronization of individual actions, which include both body actions and gaze actions, from each single object.

Next, for modeling the spatiotemporal structures of gaze actions with AHV, multiple objects' 3D gaze information is required. Thus we propose our novel 3D non-constrained and non-contact gaze estimation method that senses gaze action of the 3D video object, making full use of the multi-view video data. Our method enables to estimate the symmetry plane of the object's 3D face, based on which we can refine the original 3D shape into higher accuracy. Then by introducing the super-resolution technique, we can successfully generate virtual frontal face images of the object and perform gaze estimation on the object using a 3D eyeball model.

Finally, we propose a three-step action editing processing scheme, including data segmentation, intra-key segment editing and inter-key segment optimization. Experiment results prove that our executable 3D action editing scheme can successfully perform the synthesis of natural looking multi-party interaction 3D video scenes from separately cap-

tured data, while preserving the original action dynamics as much as possible.

Acknowledgement

I would like to express my sincerest gratitude to my supervisor, Prof. Takashi Matsuyama. He granted me the opportunity to study at Kyoto University, and provided insightful advice and constant encouragement in these five years.

I also wish to express my gratitude to other members of my thesis committee, Prof. Michihiko Minoh and Associate Prof. Atsushi Nakazawa, for taking time to review my thesis.

I would like to thank Lecturer Shohei Nobuhara for making a lot of valuable comments and supporting me to accomplish my thesis. I also would like to thank all members of Matsuyama laboratory, especially my colleagues Ryo Yonetani, Roman Humanes Pablo and Cuicui Zhang. And many thanks to secretaries in Matsuyama Laboratory, for their kind assistance.

Finally, I wish to express my deepest thanks to my family who has always supported me.

Contents

Chapter 1

Introduction

In this chapter we give a brief introduction of the research work in this thesis. First we introduce our research background, 3D video, by presenting its definition, differences from other 3D medias, and processing scheme. Then we specify the technical problem and the difficulties of this research. Next we introduce the main idea of our research work. Finally we present the outline of the following chapters in this thesis.

1.1 Background

1.1.1 What is 3D video

In the past decades, 3D video [1] has been developed as a new kind of visual information medium. 3D video is the full 3D image media that records dynamic visual events in the real world as is. It records time varying full 3D object shape with high fidelity surface properties (i.e. color and texture). Its applications cover wide varieties of personal and social human activities: entertainment (e.g. 3D game and 3D Movie), education (e.g. 3D tele-teaching system), sports (e.g. athlete technique analysis), medicine (e.g. 3D surgery monitoring), culture (e.g. 3D archive of traditional martial art), and so on.

The concept of 3D video was originated by S. Moezzi *et al.* [1] and T. Kanade *et al.* [2]. They demonstrated that full 3D human actions could be generated from multi-view video data. Many computer vision researchers have followed them to explore 3D video production technologies and applications. Nowadays several venture companies [3] are commercially supporting 3D video capture systems and 3D video content generation.

Generally, 3D video production is based on the multi-view video capture system. Since multi-view video data records 3D surface shape and texture of object(s) in motion, a temporal sequence of 3D textured mesh data is usually employed as the data structure for

Table 1.1: Functional differences between image-based and model-based methods.

Processing method	No. of cameras	Image quality	Imaged objects	Viewpoint for image rendering	Lighting environment editing	Content editing	Object motion analysis
Image-Based	several tens - hundred dense arrangement	high	entire scene natural scene	constrained around cameras	NO	NO	NO
Model-Based	several - tens sparse arrangement	average	3D object	completely free	YES	YES	YES

representing a 3D video data stream.

The most important feature of 3D video is its full 3D-ness. The full 3D shape and action of a real world object including its backside is captured as is. While many 3D visual media technologies have been developed, their differences from 3D video can be characterized as follows.

3D TV and Theater: (Fig. 1.1)

3D TV and Theater data are nothing more than a pair of 2D stereo video images, from which 3D scenes are perceived in human brains. In detail, with special equipments like 3D glasses or 3D display monitors, audiences can enjoy exciting pop-up motion pictures. However, their basic limitations constrain the observing direction of the scene to be fixed. The back, left, or right sides of the scene cannot be seen interactively. Another limitation is that they do not allow interactive modifications such as edit image contents, change object locations or poses, or modify illuminations or shadows. On the other hand, with 3D video data, one can easily generate a pair of 2D stereo video data for 3D TV and Theater, with which viewing directions and zooming can be interactively changed. That is, 3D video enables us to enjoy interactive full 3D scene observations.

3D Depth (Range) Image: (Fig. 1.2)

Many off-the-shelf devices have been developed to capture 3D depth images. Some employ laser beam time-of-flight [4, 5] and others stereo-based methods [6] to measure depth values at object surface points. Usually the captured data are represented by a depth image where each pixel value denotes the depth value from the camera center to a surface point. Note that with this technology no backside surface data of the object can be obtained. In that sense, the captured data is usually called 2.5D images rather than 3D. Although 2.5D motion images can be captured in real time with modern range recorders like Kinect [7], 3D video production technologies are required to capture full 3D motion images. Ikeuchi *et al.* [8] developed a system to

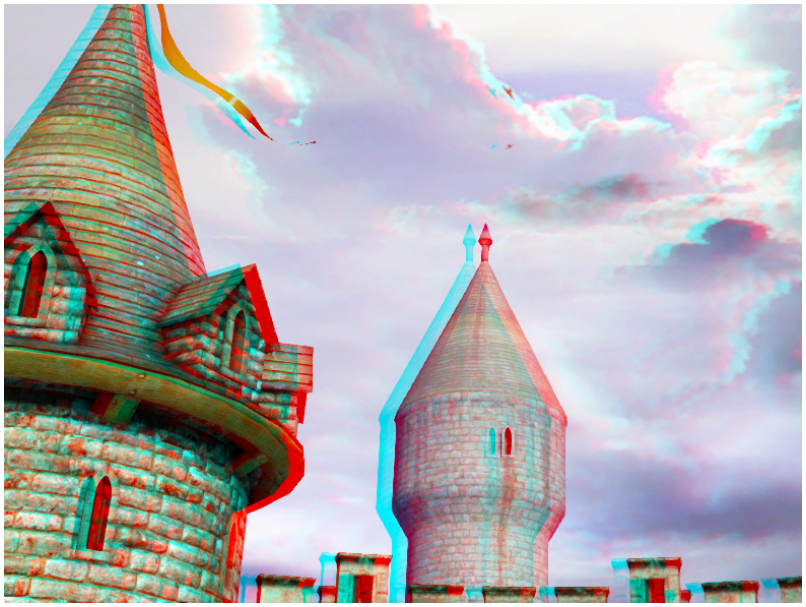


Figure 1.1: Parallax control for 3D TV.

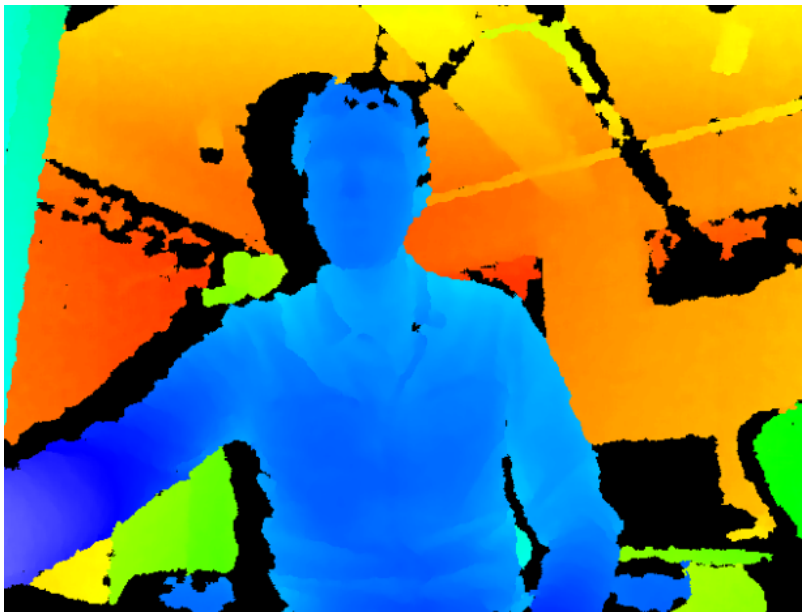


Figure 1.2: Example of 3D Depth Image generated by Kinect camera.

reconstruct full 3D shape of huge static objects like big Buddha statues and temples by patch-working a group of 2.5D images observed from different positions. Since the process to capture multi-view 2.5D images is time consuming, target 3D objects must be static ones.

3D Motion Capture [9]: (Fig. 1.3)

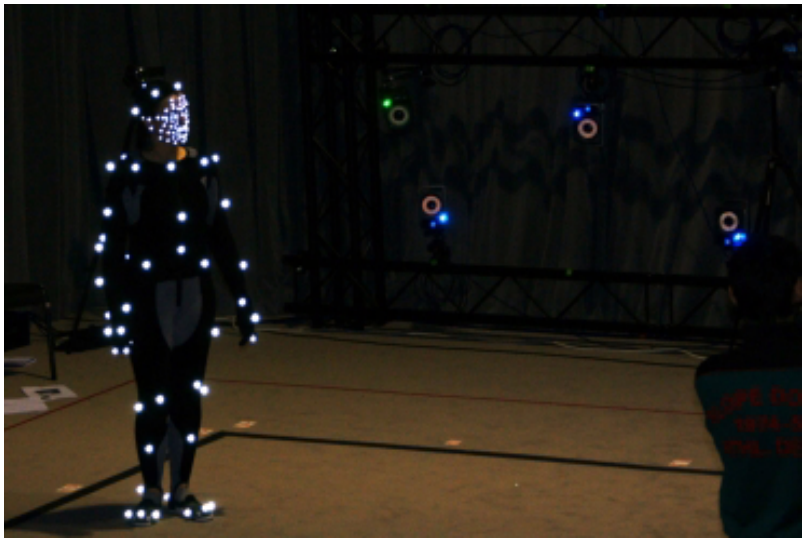


Figure 1.3: Example of 3D motion capture system. (*Vicon Motion capture system. [10]*)

3D motion capture is the process of recording the objects' 3D positions and movements. Usually markers are attached to the object surface, accompanied with a group of cameras surrounding the object. Captured data are just a group of dynamic 3D position sequences of markers, so that no 3D object surface shape, color, or texture can be obtained. 3D motion capture enables us to capture the 3D point motions, with which one can easily compute the skeletal model based motions of the object. However, 3D computer graphics technologies are still required to be employed to generate 3D motion pictures [11].

3D Animation [12]: (Fig. 1.4)

The most significant difference between 3D video and 3D animation is that: the former records natural real world objects while the latter generates designed artificial ones. Although 3D CG technologies enable us to generate very sophisticated object shapes, motions, and surface textures, they still stay at natural-looking level. From a technical point of view, while both 3D video and 3D animation often share 3D mesh data for representing 3D object surface shape, they differ in the following aspects:

1. 3D mesh data of 3D video is unstructured and can change largely from frame to frame since the 3D mesh data of each frame are independently reconstructed from a set of multi-view images. It should be noted that the quality of the reconstructed 3D data can be influenced by the object's postures (self occlusion), the calibration errors and reconstruction errors [48]. While a temporal

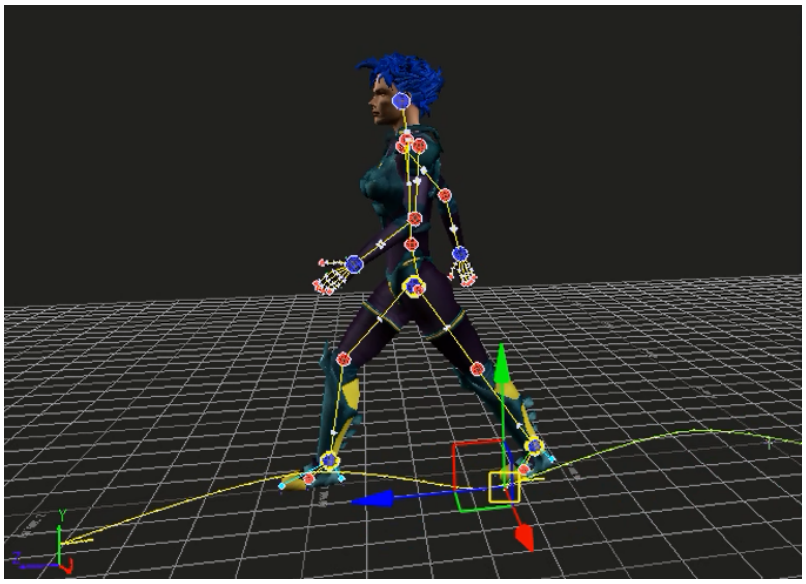


Figure 1.4: Example of 3D animation editing.

sequence of 3D mesh data in 3D animation are generated based on the pre-specified 3D motion data keeping the mesh structure (number of vertices and their connectivities) over time.

2. In 3D video, the 3D object motion from a temporal sequence of reconstructed 3D mesh data requires to be estimated, whereas in 3D animation the 3D object motion data (e.g. skeletal animations) are specified at the designing step.
3. In 3D video, surface texture and color properties of each mesh face should be estimated from observed multi-view images, while those of 3D animation are given at its designing process.

3D CT Image [13]:

A 3D CT scan, or a three-dimensional computerized tomography scan, is a type of x-ray that allows high quality images of organs, blood vessels, and bones to be recorded in a very short amount of time. The individual CT images are then layered together to generate 3D CT image that represents the virtual model of the body. 3D CT images record the 3D feature distribution in the interior volume of the object, whereas 3D video represents 3D surface shape and texture alone. Even though 3D CT image may be called substantially full 3D image, it is still difficult to capture objects in motion.

Free-Viewpoint TV [14, 15]: (Fig. 1.5)

This technology shares many features in common with 3D Video. In particular,

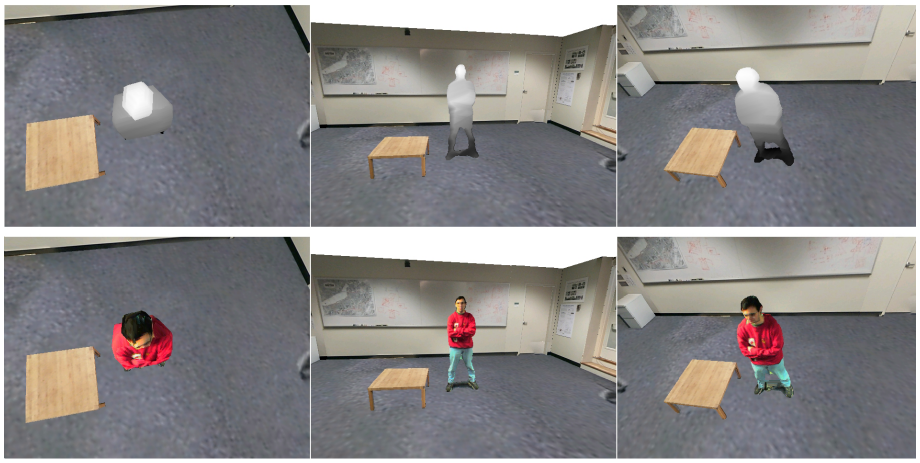


Figure 1.5: Example of Free-viewpoint image. (*Copyright 2000 ACM* [19].)

in capturing and visualizing image data: both employ a multi-view camera system where a group of cameras are set surrounding an object, and can interactively generate 2D and 2.5D object images viewed from arbitrary viewpoints. Their differences lie in data representation and processing methods: the former employs image-based methods while the latter employs model-based ones. In the image-based method, a free-viewpoint 2D image of the scene is synthesized based on geometric relations among pixels in a set of multi-view images. To represent the geometric relations, the epipolar image [16], rayspace [17], and light filed [18] representations of the multi-view images were developed. The model-based method, on the other hand, reconstructs explicitly 3D object shape and motion to render free-viewpoint images. Table 1.1 summarizes functional differences between them, from which we believe 3D video has much larger flexibilities. In 3D video, object position, shape, and motion can be analyzed and edited and lighting can be modified to render object images. Moreover, with the 3D gaze sensing method proposed in Chapter 4, we can render the first-person-view 3D video: for example, dancing actions can be viewed from the dancer's own viewpoint.

1.1.2 Processing scheme of 3D video production

The processes aligned on the left side in Figure 1.6 illustrate the basic processes to produce a 3D video frame:

(1) Synchronized Multi-View Image Acquisition:

A set of multi-view images of an object are simultaneously taken by a group of distributed video cameras surrounding the object. The top row images in Figure 1.6

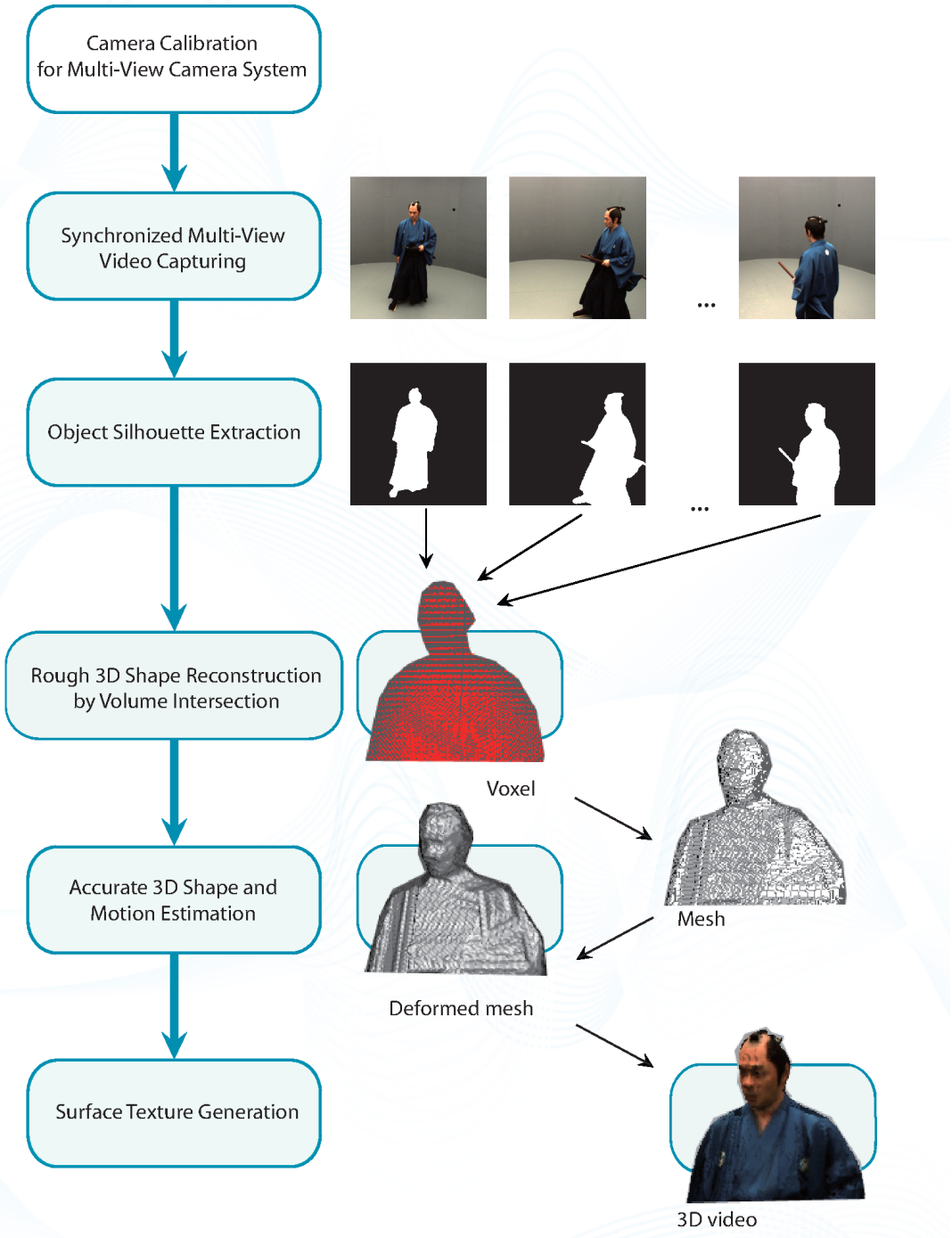


Figure 1.6: Process of 3D video production.

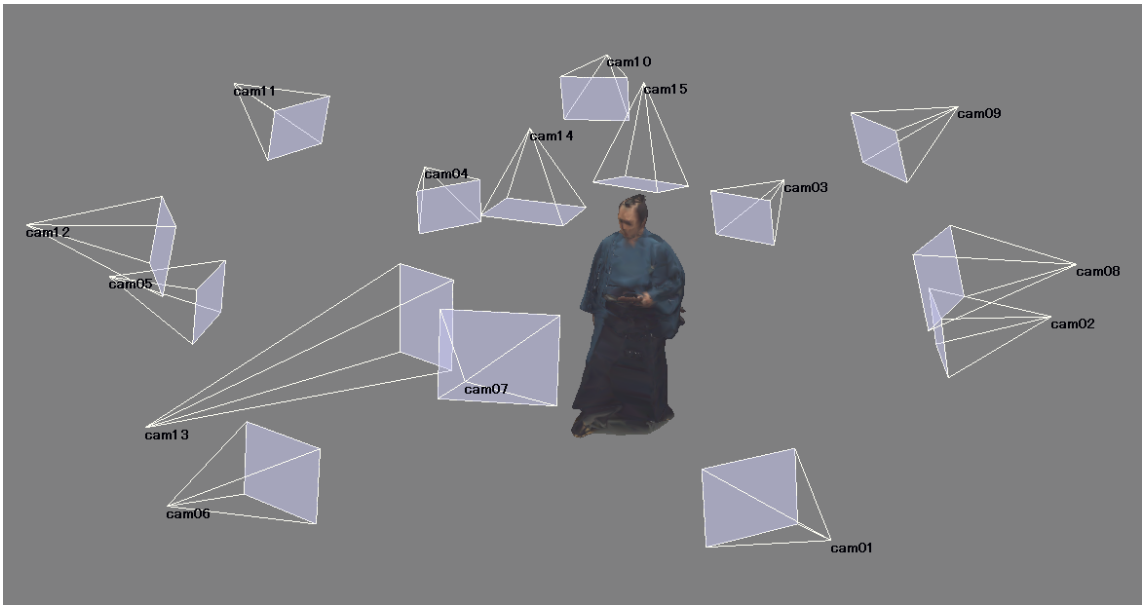


Figure 1.7: Example of multi-view video capture setup.

show samples of captured multi-view images of a traditional Japanese warrior.

In general, the cameras are spaced uniformly around the object(s) so that the captured images cover the entire object surface (Figure 1.7). In practice it is often difficult to satisfy this requirement of full observation coverage of the object surface. Some parts of the surface are occluded by others even when capturing a single object. Moreover, heavy occlusions become inevitable when capturing multiple objects simultaneously in action. Thus in order to produce a 3D video, methods that cope with self and mutual occlusions have to be developed.

Following the 3D video studio design, an important technical issue to be solved before obtaining multi-view video data is the geometric and photometric camera calibration [48]. The geometric camera calibration is the process that estimates parameters of the geometric transformation conducted by a camera, which projects a 3D point onto the 2D image plane of the camera. In general, the geometric camera calibration estimates the intrinsic, extrinsic and lens distortion parameters by observing some reference objects in the scene. On the other hand, the light flux has photometric characteristics such as colors (i.e. wave length of light) and powers (i.e. irradiance), which are also transformed through the imaging process. The process of the photometric camera calibration is to rectify the photometric transformations by a camera.

(2) Object Silhouette Extraction:

A silhouette gives an outline of a person or an object. When extracting silhouettes from an image or video, we acquire information about where in the image the foreground objects are located, their sizes and their shapes. This information is required by the model-based method in each observed image. In a well designed 3D video studio, background subtraction and/or chroma-key method can be applied to generate a set of multi-view object silhouettes (second row images from top in Figure 1.6).

Although it is more ideal to capture 3D videos in any natural environments, their geometric, photometric, and dynamical complexities forced us to work in well controlled studio environments instead. Note that by introducing depth sensing devices like TOF cameras, we may enable the 3D video system to work under natural environments with cluttered backgrounds.

(3) 3D Object Shape Reconstruction:

Up to present many methods for 3D shape reconstruction from a set of multi-view images have been developed. Among them the volume intersection remains the most simple and popular one. Each object silhouette is back-projected into the 3D world coordinate system to generate a 3D visual cone encasing the 3D object. Then, such 3D cones are intersected with each other to generate the voxel representation of the object shape (third picture from top in Figure 1.6). Since this method utilizes only silhouette information, many concave parts of the object cannot be reconstructed. Hence a 3D shape refinement process based on surface texture and motion information has to be performed. For such surface-based processing, voxel data have to be converted into 3D surface mesh data (third picture from bottom in Figure 1.6). The second picture from the bottom in Figure 1.6 illustrates a refined 3D shape obtained with the 3D mesh deformation method.

(4) Surface Texture Generation:

With the reconstructed 3D object shape, color and texture on each 3D mesh face can be computed from the original captured multi-view images (bottom picture in Figure 1.6).

By repeating the above processes for each video frame, the corresponding 3D video data allowing free viewpoint browsing can be created, as shown in Figure 1.8.

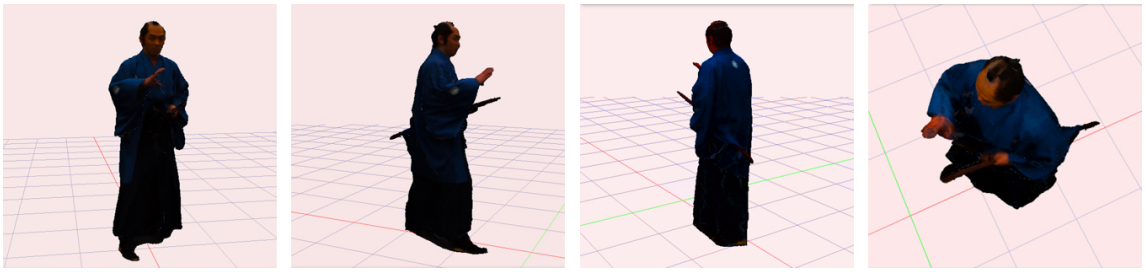


Figure 1.8: 3D video data for free viewpoint browsing.

1.2 Multi-party Interaction 3D Video Scene Synthesis: Problem Specification

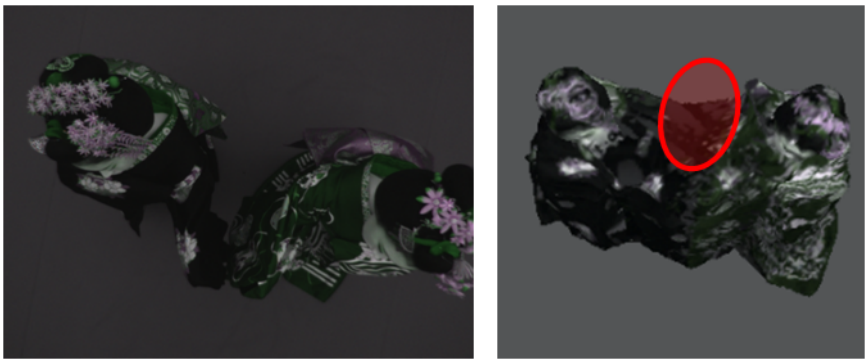
In general, the multi-view object observation for 3D video production should satisfy the following basic requirements [49]:

- **Requirement 1**: Accurate camera calibration,
- **Requirement 2**: Full visual coverage of the object surface, and
- **Requirement 3**: High spatial image resolution.

Here the second requirement means that every point on the object surface must be observed by at least two cameras to estimate its 3D position by shape-from-stereo methods. Usually, this requirement is difficult to be satisfied for an object in action. Objects performing complex actions like Yoga or wearing loose clothes like KIMONO introduce heavy self-occlusions, which prevent object surface areas from being observed by multi-view cameras. Moreover, action scenes by multiple performers continuously introduce mutual occlusions, which significantly limit the observability.

According to the processing scheme of 3D video described in Section 1.1.2, in principle multiple objects in action can be captured at the same time. In practice, however, since their mutual occlusions will inevitably occur and therefore degrade the quality of 3D video data (e.g. invisible surfaces without textures, phantom volumes (Fig. 1.9)), most of 3D video data are produced for a single object. Thus, in the acquisition and reconstruction of multi-party interaction scenes in 3D video, multiple objects are usually required to be captured separately and then, synthesized together afterwards (Fig. 1.10).

Since the separate capture scheme will inevitably result in spatial and temporal mismatches of the interaction event, the synthesis of multi-party interaction scenes from them becomes a non-trivial task and requires large amount of editing work to perform spatiotemporal alignment of the unsynchronized data sequences.



(a) Original captured image. (b) Reconstructed 3D video.

Figure 1.9: Phantom volume caused by mutual occlusion.

The core technical problems we address for this research work are (1) how we can model multi-party interaction events into spatiotemporal constraints, and (2) how we can perform spatiotemporal alignments on separately captured data given as sequences of 3D meshes. The difficulties of our technical problem mainly lie in:

(1) Spatiotemporal Unsynchronization:

As one can imagine, the separate capture scheme will inevitably result in many spatial and temporal mismatches between multiple objects' actions. In fact, many complicated multi-party interaction scenes (e.g. fierce fighting scenes from action movies) are not easy to be conducted properly even under the situation that multiple objects are performing together, especially when the duration of the interaction sequence is rather long. Performing multi-party interaction event independently by each single actor can be much more difficult, since he/she has no reference about how to cooperatively adjust his/her positions, postures and moving speeds to match other objects' actions. In order to synthesize integrated multi-party interaction scenes from these separately captured data, effective spatiotemporal alignment techniques need to be developed to solve the spatiotemporal unsynchronization among multiple objects' actions.

(2) View-independent Fidelity Consistency Requirement

As a free viewpoint browsing media, 3D video requires view-independent fidelity consistency of the editing result, meaning that the synthesized multi-party interaction scenes should be smooth and natural looking from whichever viewpoint the audiences may take. The trick of designing specific camera work to disguise the unnatural fact, which is widely used in 2D movie industries (Fig. 1.11), is no longer

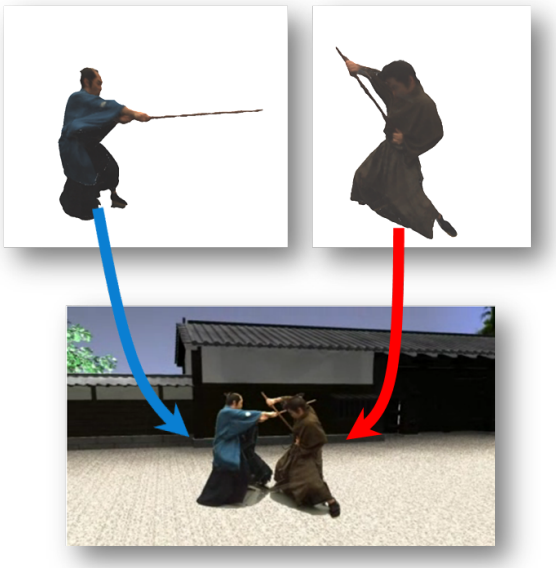


Figure 1.10: Multi-party interaction scene synthesis from separately captured data.

applicable to the synthesis work of multi-party interaction 3D video scenes.

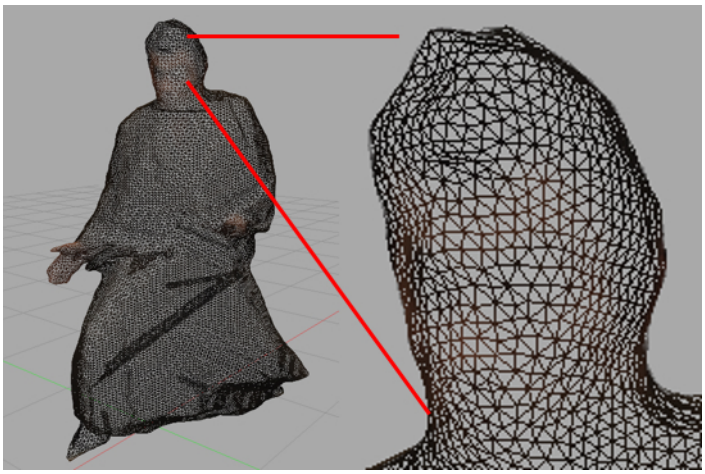
(3) Non-temporal-correspondent Data Structure

While 3D video has the advantage in capturing the realistic detailed non-rigid surface shape dynamics of the body, clothing or even hairs, its acquisition scheme results in an unstructured volumetric or mesh approximation of the surface shape at each frame without temporal correspondence (Figure 1.12). Although some research work [50][51][52][53][54] tried to estimate a smooth transition among these unstructured data, estimating accurate dense correspondence of dynamic surfaces remains an open problem. As a result, most conventional skeleton model based action editing methods are not directly applicable to the 3D video data, making the editing work of 3D video becomes more challenging than the conventional skeletal CG data or motion capture data.

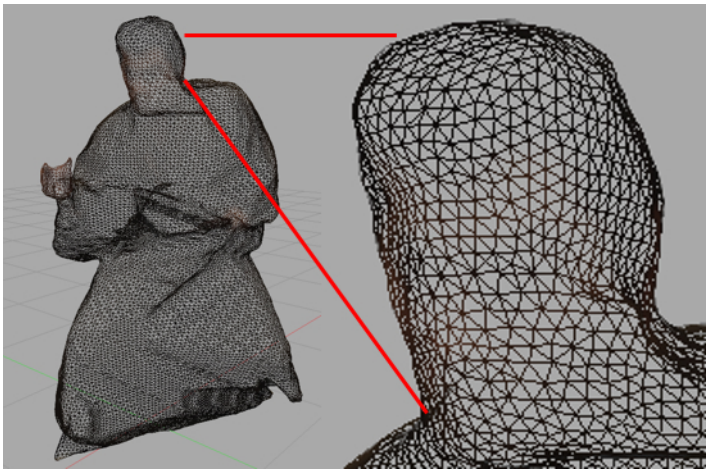
Figure 1.11: The “fake action” disguise of 2D movie.(*TV Series: Gokeninzankuro*.)

It should be noted that in different research fields, the word "interaction" has many different levels of meanings. In this thesis, since our main target is to create multi-party interaction 3D video scenes of human beings, we narrow our definition of interaction on those explicit interactive events composed of spatiotemporally correlated human actions that performed by multiple objects. Those subtle interactive factors like facial expressions are out of the concern of this thesis.

On the other hand, in addition to the spatiotemporal action editing, 3D video also allows photometric editing work. With the lighting environment estimation method proposed by Takai *et al.* [20], object surface reflectance properties can be estimated as intrinsic color and degree of secularity. In that sense, 3D video data taken under fixed illumination setup can then be visualized under arbitrary lighting environments. Theoretically, for multi-party interaction scene synthesis, recomputing the photometric features on multiple objects’ surfaces could make the synthesized scene more natural looking. However, in this thesis we locate our main focus on the spatiotemporal action editing task, and leave the photometric editing process as one of our future works.



(a) Frame t.



(b) Frame t+1.

Figure 1.12: Unstructured 3D video mesh data.

1.3 Action History Volume based Multi-party Interaction Modeling and Editing

To solve the technical problems defined in Section 1.2, we propose a novel method to synthesize spatiotemporally synchronized multi-party interaction scenes in 3D video from separately captured data, while preserving the original action dynamics of each object as much as possible. In the following subsections we first present our basic idea of modeling multi-party interaction event with both body action and gaze action. We then give a brief introduction of our three-step multi-party interaction 3D video editing strategy.

1.3.1 Modeling multi-party interaction with body and gaze actions

The main targets of the objective multi-party interaction 3D video scenes are human beings. And based on the observation of multi-party interaction 3D video scenes generated by simply put together the separately captured data, we have found out that the unnatural mismatches mainly lie in the synchronization of multiple objects' body actions and gaze actions. Thus we take these two aspects of human actions into consideration in modeling the multi-party interaction event:

(1) Spatiotemporal synchronization of body actions:

An important aspect worth concerning is the body actions of multiple objects. In multi-party interaction events, objects usually perform various body actions to directly interact with others, or implicitly convey their feelings or opinions. These body actions are spatiotemporally correlated with each other and naturally, the synchronization of body actions from different objects can be used to represent the interaction event. We introduce the Action History Volume to model single object's body actions, and build up a "multi-party interaction dictionary" based on the AHV representation in Chapter 3. Then in Chapter 5, we use that interaction dictionary to generate spatiotemporal constraints for different types of synchronized body actions in performing the editing work of 3D video.

(2) Multiple objects' gaze actions:

As the proverb "The eye is the window of the heart" says, object's gaze action is very important in multi-party interaction events since it expresses emotional feelings and reveals mental status like personal interests, attentions or concentrations. Generally, objects participate in one interaction event tend to keep their interactive partners/opponents inside their eye sights. Based on this observation, in the editing work of multi-party 3D video we introduce a "mutual visibility constraint" that requires the objects from an interaction event to be visible for each other. In Chapter 4 we discuss how we can perform 3D gaze sensing on the 3D video data using the symmetry prior and super-resolution technique, for modeling the spatiotemporal structure of objects' gaze actions using Action History Volume.

Taking into account these two important aspects, we propose the Action History Volume to describe the spatiotemporal structures of both body actions and gaze actions. Then we can model the multi-party interaction event by representing the synchronization between multiple AHVs, which encode the spatiotemporal information of object's body action or gaze action. We will detailedly discuss this issue in Chapter 3.

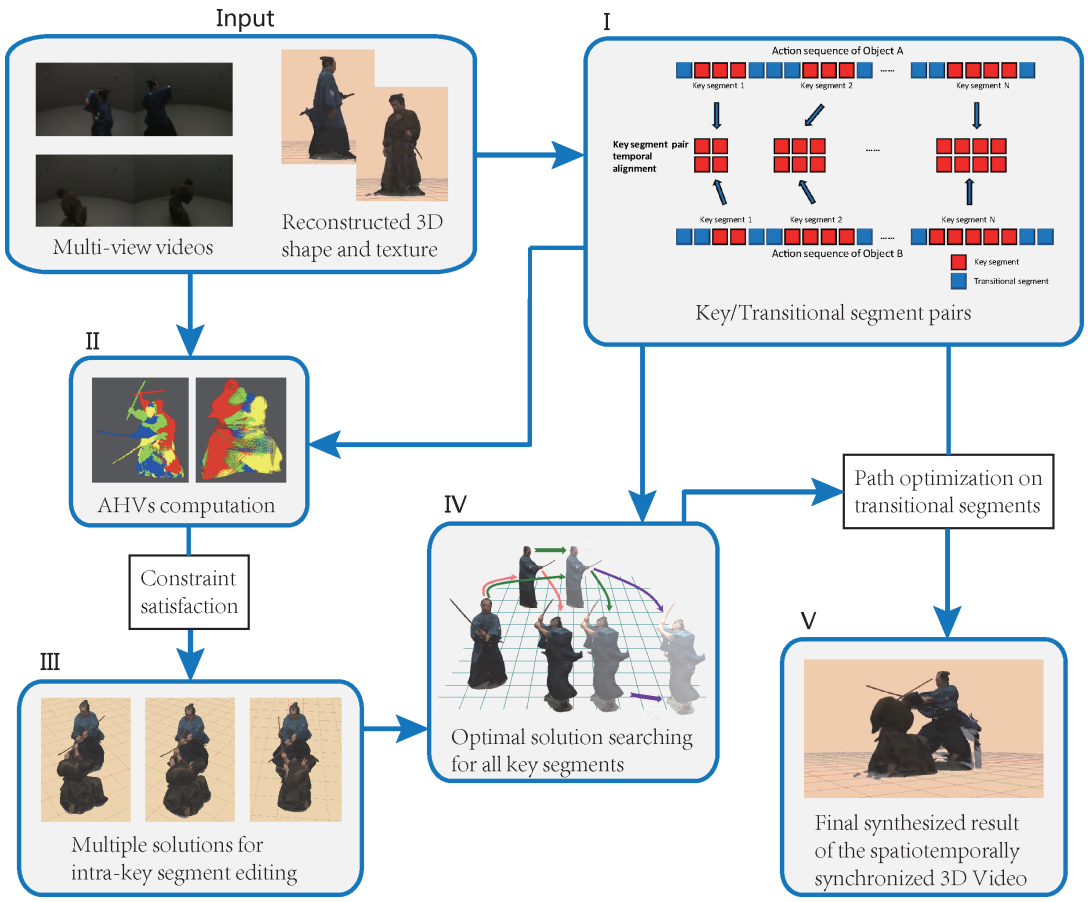


Figure 1.13: Computational processes for multi-party interaction editing. (*Copyright 2014 IEEE.*)

1.3.2 Constraint based three-step 3D video editing strategy

In order to perform an effective multi-party interaction 3D video scene synthesis, we propose a constraint based three-step 3D video editing strategy, whose input and output are:

Input:

(1) Temporal sequences of 3D video mesh data of multiple objects performing pre-designed interactive actions, captured and reconstructed separately.

(2) The corresponding multi-view video image data of multiple objects.

Output:

The 3D transitions for each frame of each object's 3D video data that synthesize

Figure 1.14: Multi-party interaction sequence containing multiple short local interaction events.

them as natural looking multi-party interaction scenes when they are put together into the same world coordinate system. The 3D transitions includes:

(1) Translation parameters that relocate the 3D video mesh data on the ground plane (xy-plane).

(2) Rotation parameters that rotate the 3D video mesh data around the gravity axis (z-axis) passing through its center of mass.

The target multi-party interaction 3D video in this research can be a long sequence containing many short local interaction events (Figure 1.14). Usually the spatiotemporal mismatches for each short local interaction event are different, so it is nearly impossible to solve the unsynchronization problem at once by simply performing a unified global adjustment on the whole sequence. Instead, since those short local interaction events are usually independent from each other, a natural idea is to handle the short local interaction events separately and combine them together afterwards. Based on this concept, we design a three-step editing strategy as follows:

(1) Data segmentation:

We first divide the original captured action sequences into key segments and transitional segments, based on whether multiple objects are contacting with each other or not during the interaction event. Since the original data of each object are supposed to match with each other, these key segments and transitional segments should be in pairs. And for each pair of key/transitional segments, a temporal alignment will be performed to unify their temporal duration.

(2) Intra-key segment editing:

By introducing Action History Volume (AHV) we model the synchronization of multiple objects' body actions and gaze actions into spatiotemporal constraints, and

use them to compute the proper relative positions of multiple objects for each pair of key segments. Note that for each key segment pair there can exist multiple solutions.

(3) Inter-key segment optimization:

Finally, a global optimization will be performed to combine the whole interaction sequence together while keeping it smooth and preserving the original actions. Note that the proposed method does not require explicit surface correspondence across time, and does not rely on any skeletal based editing technique.

Fig. 1.13 illustrates an overview of the computational processes of the proposed method. Given a pair of sequence of 3D mesh data and corresponding multi-view video (Fig. 1.13 Input), the proposed multi-party interaction editing method applies the following processes: (1) first perform data segmentation and temporal alignment to divide the original action sequences into pairs of key segment and transitional segment (Fig. 1.13 I). Then (2) compute the AHVs of each object's body action and gaze action for each key segment pair (Fig. 1.13 II). Next, (3) for each key segment pair estimate the proper relative positions of multiple objects by an AHV-based constraint satisfaction (Fig. 1.13 III). After that, (4) select the optimal solution for each key segment pair, so that the combination of them forms up the optimal solution of the whole synthesized multi-party interaction sequence, minimizing the artifacts generated by the editing work (Fig. 1.13 IV). Finally, (5) estimate the optimal paths for all the transitional segments to achieve a synthesized multi-party interaction 3D video result that preserves the natural action dynamics of the original data (Fig. 1.13 V).

Here Process I equals to the Data segmentation step, Process II and III belong to the Intra-key segment editing step, and Process IV and V form up the Inter-key segment optimization step.

It should be noted that process I, data segmentation, serves as a pre-processing of the whole action editing work. Since it is out of the scope of this thesis, this step is assumed to be performed manually by the editor. Step II to V can be automatically computed using our action editing algorithms presented in Chapter 5.

The main contribution of this thesis consists of (1) we propose a concept of Action History Volume, as well as an AHV-based Interaction Dictionary that enables to model all types of multi-party interaction events; (2) we propose a novel 3D non-constrained and non-contact gaze estimation method based on symmetry prior; and (3) we have designed an executable computational scheme for performing the spatiotemporal editing of real multi-party interaction 3D video. Possible applications include the making and editing of 3D movies, computer animations and video games.

1.4 Outline

The rest of this thesis is organized as follows. First, we review related studies to clarify the novelty of this work in Chapter 2. We then introduce the AHV-based multi-party interaction modeling method in Chapter 3, our novel 3D gaze sensing method for gaze action modeling in Chapter 4, and detailed description of our spatiotemporal 3D editing algorithm with evaluation in Chapter 5. Finally, we summarize the proposed method with discussions on future work in Chapter 6.

Chapter 2

Related Work

In this chapter, we review the conventional researches related to the work of this thesis. In the following sections, we first give a brief introduction of conventional action editing researches for both skeletal data and 3D video data, accompanied with a discussion on classic action representation approaches. We then recall the traditional gaze estimation works to clarify the novelty and contribution of our 3D gaze estimation method that takes advantage of the symmetry prior.

2.1 Action Editing Research Work

2.1.1 Action editing work of skeletal data

Marker-based skeletal performance capture techniques have been introduced to the entertainment industry for over 30 years [59][60]. The past decade has witnessed an explosion of new techniques supporting various action editing tasks, as well as deployment of commercial tools.

In conventional studies, many researchers have followed the traditional process invented by the animators: first, edit a set of key-frames of the sequence, creating a set of poses that satisfies the constraints set by the user; and second, create in-between poses that preserve the naturalness of the original action. Gleicher [61] and Lee and Shin [62] have proposed space-time editing approaches that perform interactive action editing via key-frame manipulation. Gleicher solves for both space and time constraints simultaneously. Lee and Shin modify the poses of the skeleton in the key-frames by means of an inverse kinematics solver (IK), and then apply a multilevel B-Spline approximation for the interpolation of poses. Besides, Brundelin and Williams [63] introduced parametric motion control by interpolating pairs of skeletal motions. Parametric motion synthesis

was extended to blending multiple examples to create a parameterized skeletal motion space [64][65][66][67]. This allows continuous interactive motion control through high-level parameters such as velocity for walking or hand position for reaching. In addition, Heck and Gleicher [68] introduced parametric motion graphs combining skeletal motion parameterizations with motion graphs to allow interactive character animation for a wide range of actions with high-level continuous control.

On the other hand, Zordan *et al.* [69] have introduced a technique for incorporating unexpected impacts into a motion capture-driven animation system through the combination of a physical simulation and a specialized search strategy. Ye *et al.* [70] have proposed a fully automatic method that learns a nonlinear probabilistic model of dynamic responses from very few perturbed sequences. Their model is able to synthesize responses and recovery actions under new perturbations different from those in the training examples.

Although these approaches have enabled flexible editing and reuse of skeletal motion capture for character animation, they are not directly applicable onto the 3D video data. Embedding the skeletal model for the 3D video sequence itself can be a tough work, since a unified and structured mesh model is not originally available. And for those 3D video objects wearing complicated costumes, accurately estimating the skeletal structure is nearly impossible with the existing techniques. Moreover, skeletal model based editing methods consider only the skeletal poses without retaining the detail of captured surface dynamics. Using these methods to perform 3D video editing will inevitably destroy the naturalness of the original captured data.

2.1.2 Action editing work of 3D video data

As is stated in the introduction, the 3D video data lacks temporal coherence in the mesh sequence, and this fact has prohibited the development of straightforward methods for manipulation. Hence, conventional editing algorithms of unstructured 3D video only focus on rearranging the original sequence into a series of sub-sequences, by finding smooth transition frames between sub-sequences based on various similarity metrics.

Huang *et al.* [55] concatenated clips of captured sequences by determining transition links using similarity matrices based on shape histograms. Starck *et al.* [56] demonstrated animation from databases of mesh sequences of actor performance by concatenating segments of captured sequences. Their approaches are analogous to previous example-based approaches to concatenative synthesis used for 2D video [71][72][73] and motion graphs used for skeletal motion capture [74, 75]. Recently, example-based approaches through re-sampling video sequences have been extended to body action [76][78] allowing off-line

animation via key frame or skeletal motion. Moreover, Tung and Matsuyama [77] proposed a topology-based shape descriptor dictionary, enabling progressive summarization of 3D video. Casas *et al.* [57] presented a 4D parametric motion graph representation, allowing real-time interactive character animation from actor performance capture in a multiple camera studio, while preserving the natural dynamics of the captured performance.

These approaches preserve the realism of the captured sequences in rendering but their applications are limited to single object animation. None of them enables the modeling and editing of multi-party interaction 3D video scenes. The central objective of this thesis is to present a novel spatiotemporal editing framework that synthesizes well synchronized multi-party interaction 3D video scenes from separately captured data.

2.1.3 Action representation approaches

In computer graphics and computer vision area, researchers have invented various approaches for describing actions. Positions and velocities of human body parts have been used by Green *et al.* [69] for human movement recognition. Optical flows [70], motion templates [55] and space-time volumes [56, 71] are also widely used for solving tracking and recognition problems. Such methods are mainly used for describing the object's action features for recognition tasks, and they do not offer any controllable factors for action editing task.

Besides, kinematic models [66, 72] as shown in Figure 2.1 are widely used in robotics, motion capture system and computer animation editing. They can represent various kind of actions, and as well they provide easily controllable factors for the editors. However, the kinematic models are lacking in the ability of representing multi-party interaction events. Few spatiotemporal constraints can be directly defined between multiple objects based on them. Not to mention that for certain type of unstructured data [73] without a unified mesh model, kinematic structures are even not directly applicable.

On the other hand, action representations encoding both spatial and temporal information have been invented. Bobick and Davis introduced Motion History Images (MHI) and Motion Energy Images (MEI) in [21] to capture motion information in images. They encode, respectively, where motion occurred, and the history of motion occurrences, in the image. Pixel values are therefore binary values (MEI) encoding motion occurrence at a pixel, or multiple values (MHI) encoding how recently motion occurred at a pixel. The

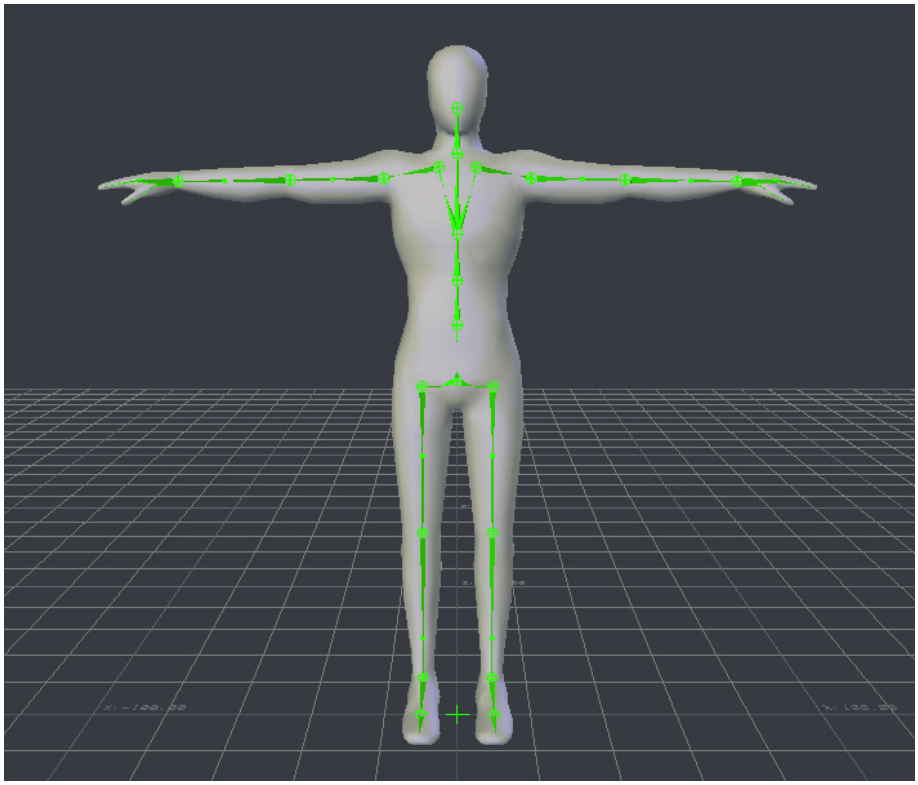


Figure 2.1: Example of kinematic models.

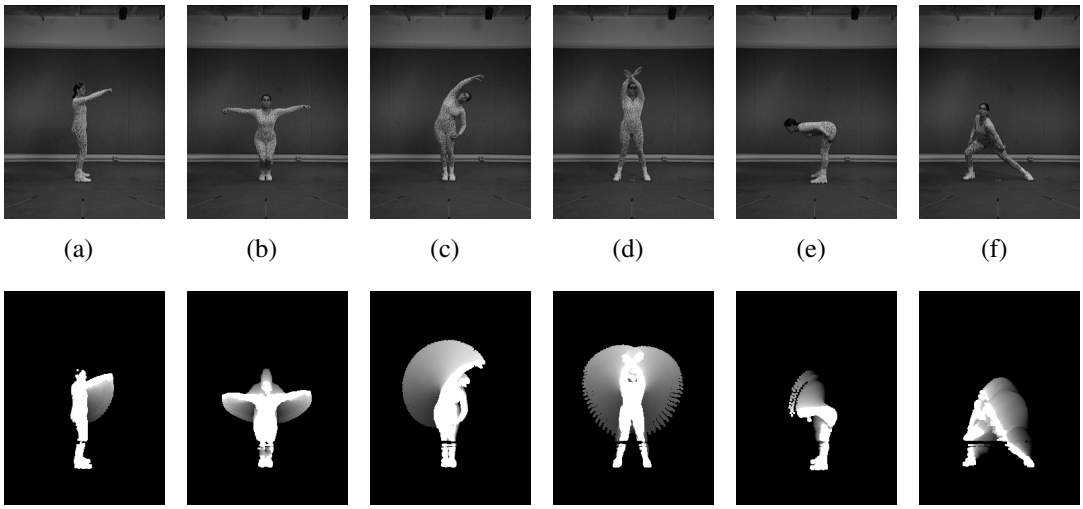
MHI function is defined by:

$$h_\tau(x,y,t) = \begin{cases} \tau & \text{if} D(x,y,t), \\ \max(0, h_\tau(x,y,t-1) - 1) & \text{otherwise.} \end{cases} \tag{2.1}$$

where τ is the maximum duration a motion is stored. The associated MEI can be easily computed by thresholding $h > 0$.

As is shown in Fig. 2.2, the MHI carries out a concept of considering the entire motion as a whole instead of describing it for each single frame, by encoding the history of motion occurrences in the volume.

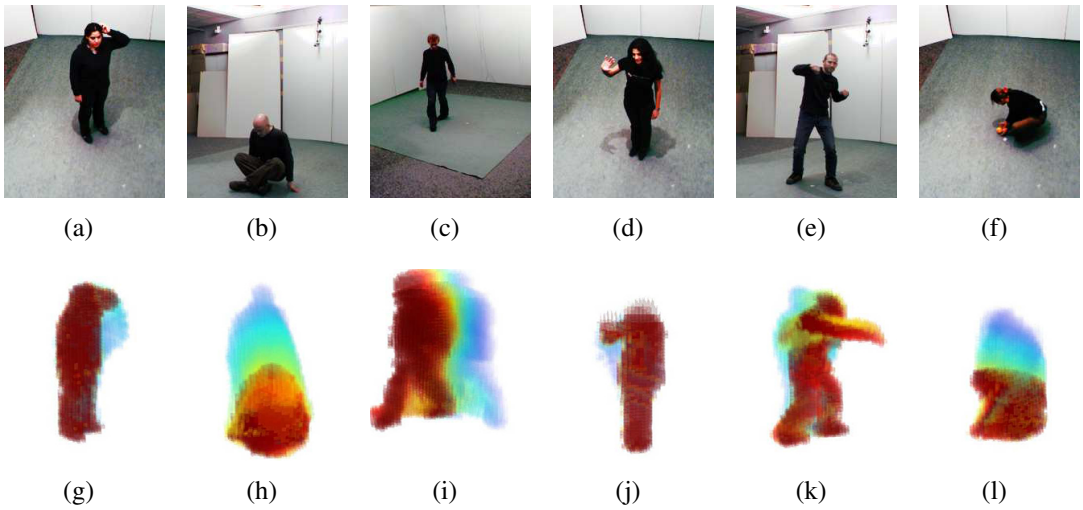
Weinland *et al.* [79] has proposed a generalization of the 2D motion templates into 3D, which they call the 3D Motion History Volume (MHV), for solving free viewpoint action recognition task. Mathematically, consider a binary-valued function $D(x,y,z,t)$ indicating motion at time t and location (x,y,z), then their MHV function is defined as



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 2.2: 2D Motion History Images (MHI) (*Copyright 2006 CVIU* [79].)



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Figure 2.3: 3D Motion History Volume (MHV) (*Copyright 2006 CVIU* [79].)

follows:

$$v_\tau(x,y,z,t) = \begin{cases} \tau & \text{if} D(x,y,z,t), \\ \max(0, v_\tau(x,y,z,t-1)-1) & \text{otherwise.} \end{cases} \tag{2.2}$$

Fig. 2.3 illustrate the MHV based action representation.

Their 3D representation exceeds the 2D motion templates in that:(1) it is more informative since additional camera calibration information is taken into account, (2) the 3D

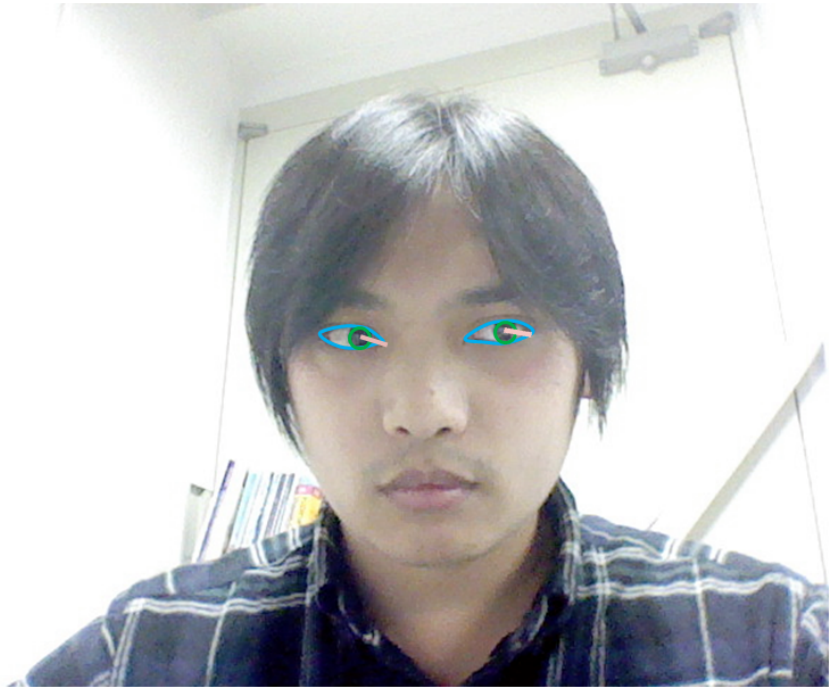


Figure 2.4: Gaze estimation using frontal face image.

representation is more robust to the object's positions relative to the cameras as it replaces a possibly complex matching between learned views and the actual observations by 3D alignment, and (3) the 3D representation allows different camera configurations.

Our idea of Action History Volume is inspired by the 3D Motion History Volume. In Chapter 3 we will present our AHV and discuss how it exceeds MHV in the capability of representing multi-party interaction events.

2.2 Gaze Estimation Research Work

2.2.1 Gaze estimation from frontal face images

In gaze estimation literature, most studies rely on the 2D frontal face images, as is shown in Fig. 2.4. Yamazoe *et al.* [30] proposed to track 2D eye features from images captured by a single camera to estimate the horizontal and vertical gaze angles in 3D space. The use of Active Appearance Models(AAM)[31] and 3D eyeball model has been proposed by Ishikawa *et al.* [32]. In Guestrin *et al.* 's work[33], a single camera with multiple calibrated light sources are used for 3D gaze estimation. And Matsumoto *et al.* [34] proposed to use stereo camera to estimate the 3D eye position and 3D visual axis. Besides,

by combining image saliency with 3D eye model, Chen *et al.* [35] proposed a probabilistic gaze estimation method that requires no active personal calibration. In addition, Weiqle *et al.* 's research[23] verified the excellent effectiveness of a commercial eye-gaze tracker, Tobii.

While these works have realized effective and robust gaze estimation, they all suffer a drawback that the head motion of the object is strictly limited in a small range, making it impossible to estimate the gazing direction from a freely moving object. In order to perform effective gaze estimation on 3D video for the gaze action modeling in multi-party interaction scenes, we proposed a novel 3D gaze sensing method using symmetry prior. The main contribution of our 3D gaze estimation work is the proposition of a method that can generate virtual frontal face images and perform gaze estimation on freely moving objects from ordinary 3D video data. Once we obtained a virtual front face image of the object from multi-view videos as we have introduced above, we utilize a conventional method which generates 3D gaze direction from an image with an extension in computing global 3D gaze direction[32]. In conventional 3D gaze direction estimation from a 2D image, they require a calibration which maps apparent gaze directions to the ones in a world coordinate. Thanks to the fully-calibrated multi-view camera environment, our method can easily convert apparent gaze directions into the global world system.

Chapter 3

Action History Volume for Multi-party Interaction Modeling

In this chapter we present the idea of Action History Volume (AHV) and introduce the multi-party interaction modeling method using AHV. The purpose of this method is to model multi-party interaction events into different types of spatiotemporal synchronization between multiple objects' actions, creating constraints that can be used in future computation process of the action editing work.

In the following sections, we first introduce the definition of Action History Volume (AHV), which encodes both the spatial and temporal information of the object's action. We then explain how AHV can be used to represent both body actions and gaze actions. Finally, we present the AHV-based Interaction Dictionary, which could be used to model multi-party interaction events.

3.1 Action History Volume

3.1.1 Definition of AHV

As discussed in Chapter 2, the MHV based action representation has been proved to work effectively for action recognition. However, the original definition of MHV only considers the occurrence of the motion. Since our goal is to perform spatial and temporal alignments on multiple separately captured action sequences, full temporal information including both the starting and ending moments of the action is necessary for representing the spatiotemporal synchronization between multiple objects' actions. Therefore we propose our Action History Volume, which extends the original MHV by adding in more

temporal information of the action as Eq. (3.1):

$$v_\tau(x,y,z,t) = \begin{cases} (t_{\text{start}_1}, t_{\text{end}_1})...(t_{\text{start}_n}, t_{\text{end}_n}) & \text{if} \bigcup_{s=0}^{\tau-1} D(x,y,z,s), \\ \text{empty} & \text{otherwise.} \end{cases} \tag{3.1}$$

where t_{start_i} and t_{end_i} $(i \in (1,...,n))$ form up one pair of the starting and ending times of actions at voxel (x,y,z) within time τ, and n denotes the total number of pair $(t_{\text{start}}, t_{\text{end}})$ at voxel (x,y,z) within time τ. $D(x,y,z,s)$ is a binary-valued occupancy function judging whether voxel (x,y,z) is occupied by the object at moment s. This function is estimated using multi-view silhouettes and thus, corresponds to the visual hull, which is easy to compute and yield robust 3D representations. We denote the centroid of AHV $v_\tau(x,y,z,t)$'s spatial volume as $C(x,y,z)$, which is considered as the root node of an AHV.

It should be noted that AHV is a voxel wise representation. That is, within a period of time τ, if the object's actions occupy voxel (x,y,z) for at least one moment, we keep record of the starting and ending times of the action on that voxel. Then the object's AHV of this time period is a collection of all those time recorded voxels.

Fig. 3.1 visualizes the AHV of an object swinging his body to the his left side. Here each voxel of the AHV is color coded based on the starting and ending times of the action on it.

3.1.2 Body Action Modeling using AHV

With the proposed Action History Volume, we can represent both the spatial and temporal aspects of objects' body actions. Fig. 3.2 illustrates AHVs of the attacker and the dodger in a fighting scene, respectively. Here different colors represent different ending times of the action at the voxels. Note that AHV needs not necessarily contain the action of the entire body. Instead it can be simplified by only counting in the partial volume of interest on the object's body. For example, in a sword fighting scene we may only care about the weapon of the attacker, so that an AHV of the sword would be enough for further editing work.

It should be noted that the mathematical definition of AHV allows multiple pairs of $(t_{\text{start}}, t_{\text{end}})$ to exist simultaneously on one voxel, meaning that a single AHV can model both unidirectional and multidirectional actions. However, in order to keep the simplicity of the AHV-based body action representation, we ensure the body action in each AHV to be unidirectional through the data segmentation step in the real editing process.

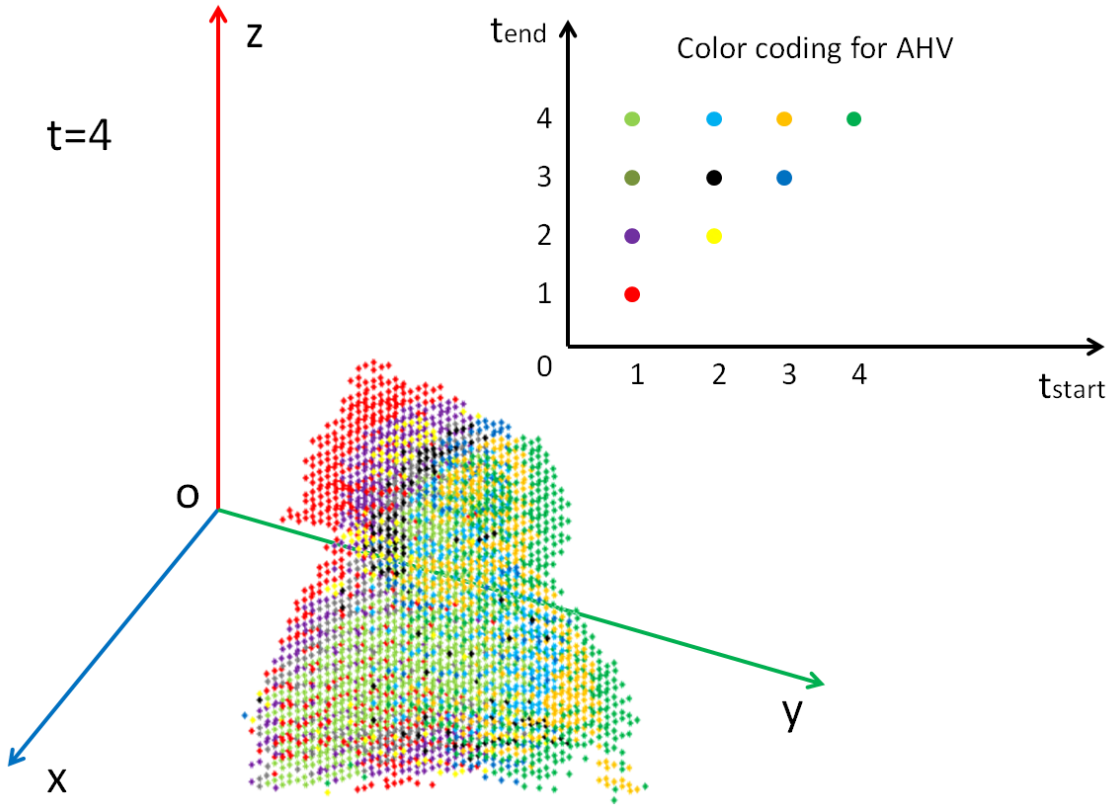


Figure 3.1: Illustration of Action History Volume. (*Copyright 2014 IEEE.*)

While in the real world human actions are usually not unidirectional, we assume that all reciprocating actions can be decomposed into unidirectional sub-actions, which can be modeled using AHVs with single pair of starting and ending times on each voxel. We ensure this requirement by taking it as one criterion in performing data segmentation, as described in subsection 5.1.2. The applicability of this assumption is justified by the evaluation in Section 5.2.

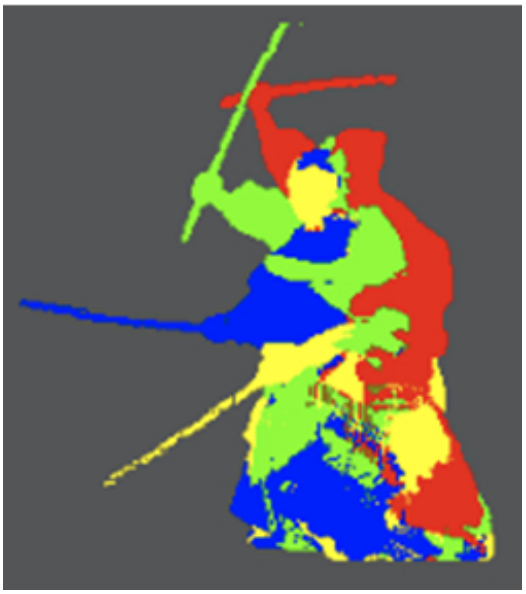
3.1.3 Gaze Action Modeling using AHV

As written in Chapter 1, objects' gaze actions need to be taken into consideration for performing multi-party interaction scene editing work. In the following subsection we explain how AHV can be used to model the object's gaze action.

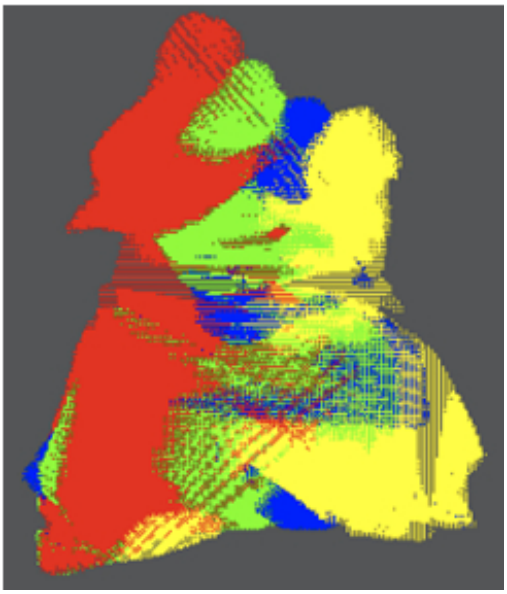
In many gaze related research works, researchers represent human being's gaze action with the gaze vector, which is the ray that runs between the center of the fovea in the retina, through the cornea to a gaze fixation point (neglecting the kappa offset between the visual and optical axis). While in this research work we propose to use the "gazing

(a) Original image of the attacker.

(b) Original image of the dodger.



(c) AHV of the attacker.



(d) AHV of the dodger.

Figure 3.2: Examples of AHV action representation.

cone" to describe the object's gaze actions, for the following two reasons:

(1) As a matter of fact, human being's eye sight covers a wide area rather than focusing on one specific gazing point through a single ray. Thus it is more natural to use a 3D volume instead of a 3D vector to describe the object's eye sight.

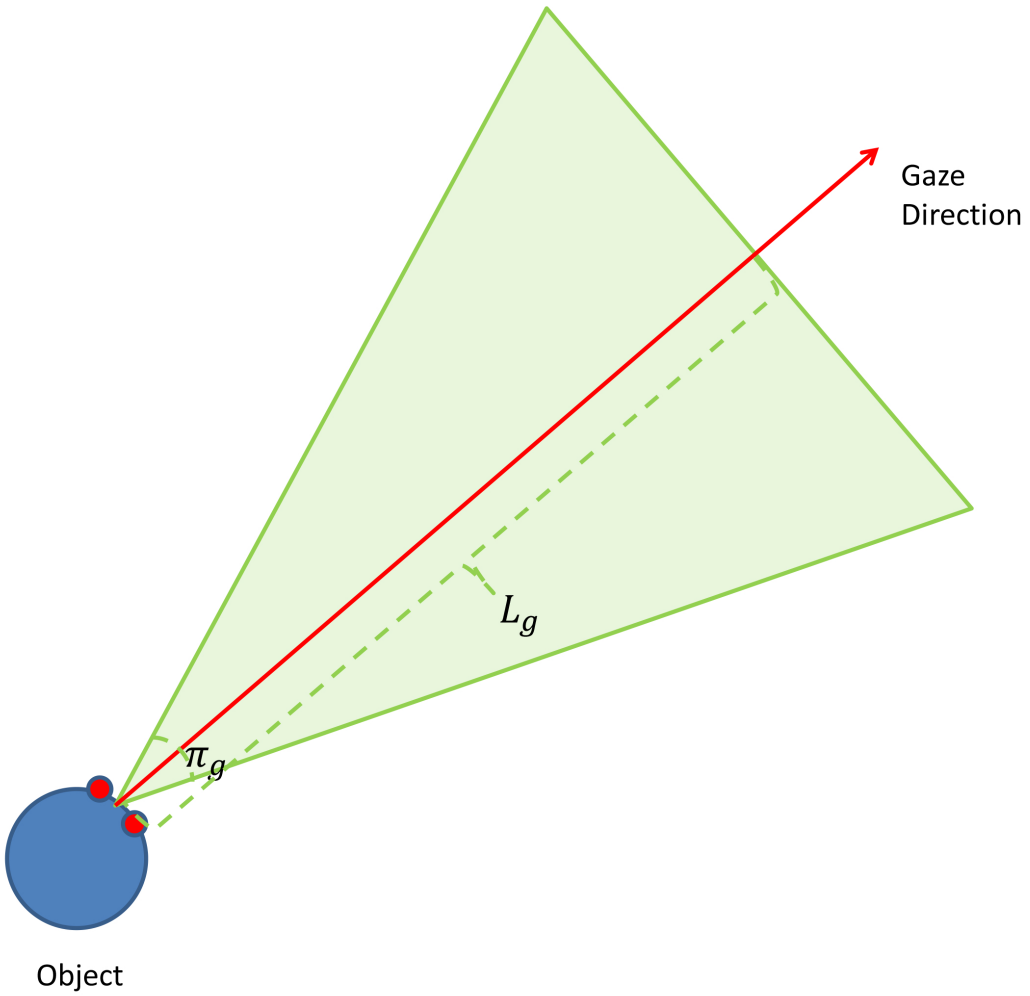


Figure 3.3: Gazing cone of object.

(2) As is written in Chapter 1, our main purpose of introducing the gaze actions is to create mutual visibility constraints that ensure interacting objects are inside each other's eye sight. Since evaluating the exact precise gazing point of each object is not really our concern, it is more suitable to use the object's field of vision instead of his/her specific gazing direction (gaze vector) to model his/her gaze actions in multi-party interaction event.

As shown in Fig. 3.3, the "gazing cone" describing the object's field of vision is a right circular cone, whose apex locates at the center of the object's two eyes, and axis equals the line on which the object's gaze vector lies. Note that π_g and L_g are changeable parameters that adjust the aperture and height of the gazing cone.

For the editing work of multi-party interaction scenes, we set π_g and L_g to be 60

degree and 5 meters respectively to specify the object's communicative visual area in interaction events, that we call an "interactive zone" of the object. And normally for most of the time, objects taking part in one interaction event should be inside each other's interactive zone. Considering the gazing cone as the interest volume, then for a temporal segment the AHV of gazing cones (interactive zones) can be built to model the spatiotemporal structures of object's gaze action.

Note that in creating the object's gazing cone, his/her gaze vector and the 3D position of his/her centroid point between the two eye centers are required. We will present in the following Chapter how we perform 3D gaze sensing on 3D video data to acquire these information.

In addition, we will discuss in Chapter 5 how these AHVs representing gaze actions can be used to describe the mutual visibility constraint in multi-party interaction events.

3.2 Multi-party Interaction Dictionary for Modeling Synchronization between AHVs

The AHV enables to describe the spatiotemporal structure of a single object's actions. In order to perform effective editing of multi-party interaction events, we propose an AHV-based method that models the spatiotemporal synchronization among multiple objects' actions.

3.2.1 AHV Surface Labeling and Multi-party Interaction Dictionary

In this research we propose to model multi-party interaction events by considering them as pairs of spatiotemporally synchronized actions performed by different objects. As mentioned in the former section, we assume that all reciprocating actions can be decomposed into unidirectional sub-actions, so that all kinds of interaction events can also be considered as pairs of spatiotemporally synchronized unidirectional actions. In that sense, the problem of modeling multi-party interaction events equals that of modeling all types of spatiotemporally synchronization between unidirectional actions, which can be modeled using simple AHVs with single pair of starting and ending times on each voxel.

The key idea is to descretize all possible interactions composed of unidirectional actions into a set of combinations of action directions. Up to this point, the processing unit of interaction modeling is assumed to be an interaction composed of purely unidirectional simple actions that can be modeled by a pair of AHVs. Hence, we can categorize AHV

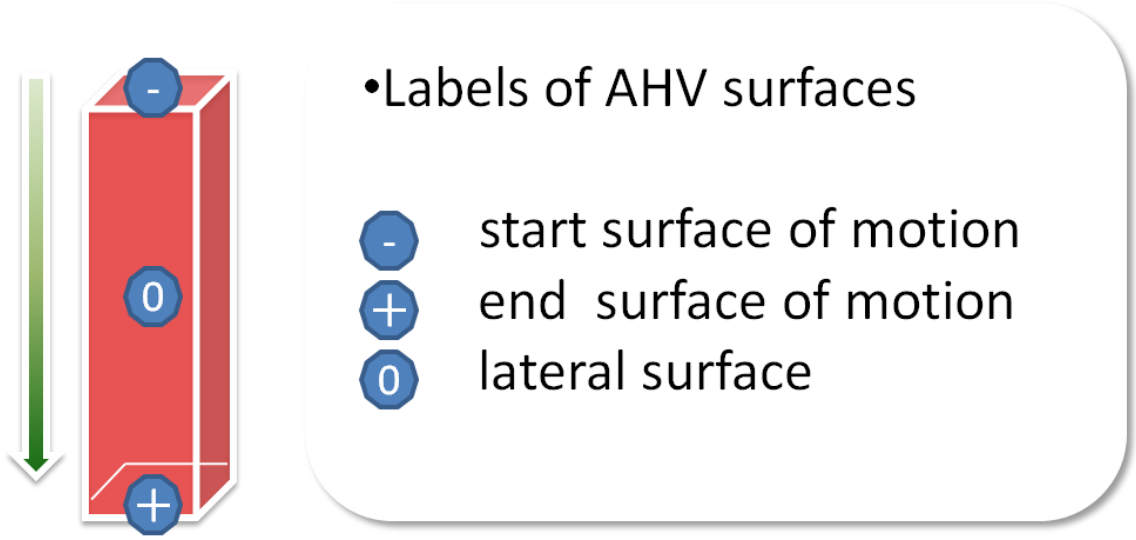


Figure 3.4: Labels of AHV surfaces. (*Copyright 2014 IEEE.*)

Table 3.1: Multi-party interaction dictionary with examples.

Interaction Dictionary	⊕ & ⊖	⓪ & ⓪	⊕ & ⊕	⊖ & ⊖	⊕ & ⓪	⊖ & ⓪
Fighting	Push&Pull	Attack&Dodge	Attack&Guard	After A&G	Punch	After Punch

surfaces into three disjoint regions where the action starts (⊕), ends (⊖), and the others (⓪), as is shown in Fig. 3.4. Their combinations describe all possible interactions between a pair of AHVs as listed in Table 3.1. We name this a multi-party interaction dictionary.

The completeness of the proposed multi-party interaction dictionary is guaranteed as follows: since multi-party interaction of reciprocating actions can be considered as a combination of sub-interactions that composed of only unidirectional actions, and the multi-party interaction dictionary enables modeling all interaction types between unidirectional actions, conclusions can be made that the proposed multi-party interaction is capable of modeling all kinds of interaction events consisting of spatiotemporally synchronized actions.

3.2.2 Mathematical Constraints for Multi-party Interaction Dictionary

In the multi-party interaction dictionary, each type of interaction contains specific spatiotemporal constraints. The mathematical descriptions of these constraints are given as follows.

First of all, for all interaction types there is a spatial constraint which states that the 3D volume of two AHVs should contact with each other:

$$V_\tau^A(x,y,z) \cap V_\tau^B(x,y,z) \neq \phi, \tag{3.2}$$

where $V_\tau^A(x,y,z)$ and $V_\tau^B(x,y,z)$ represent the collections of voxels in $v_\tau^A(x,y,z,t)$ and $v_\tau^B(x,y,z,t)$ respectively. Note that the two paired AHVs should have the same maximum duration τ (as described in Chapter 5). As well, the recorded timing should be scaled into the segment-oriented timing.

Second, each interaction type has its own temporal constraint as follows:

⊕ & ⊖:

$$\begin{gathered} \forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ t_{\text{end}}^A = T_{\text{end}} \quad t_{\text{start}}^B = T_{\text{start}}. \end{gathered} \tag{3.3}$$

⓪ & ⓪:

(a) If the interest volumes contact with each other at every moment within τ:

$$\bigcap_{t=0}^{\tau-1} (V_t'^A(x,y,z) \cap V_t'^B(x,y,z)) \neq \phi. \tag{3.4}$$

(b) If the interest volumes contact with each other at certain moments within τ:

$$\bigcup_{t=0}^{\tau-1} (V_t'^A(x,y,z) \cap V_t'^B(x,y,z)) \neq \phi. \tag{3.5}$$

(c) If the interest volumes have no contacts τ:

$$\begin{gathered} \forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ t_{\text{start}}^A > t_{\text{end}}^B. \end{gathered} \tag{3.6}$$

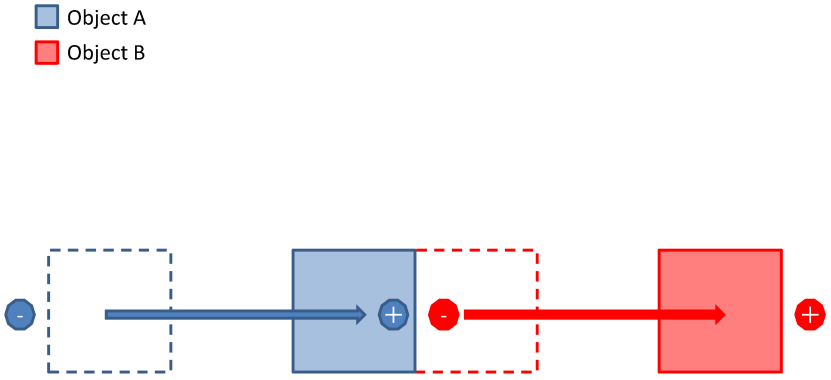


Figure 3.5: Interaction type ⊕ & ⊖.

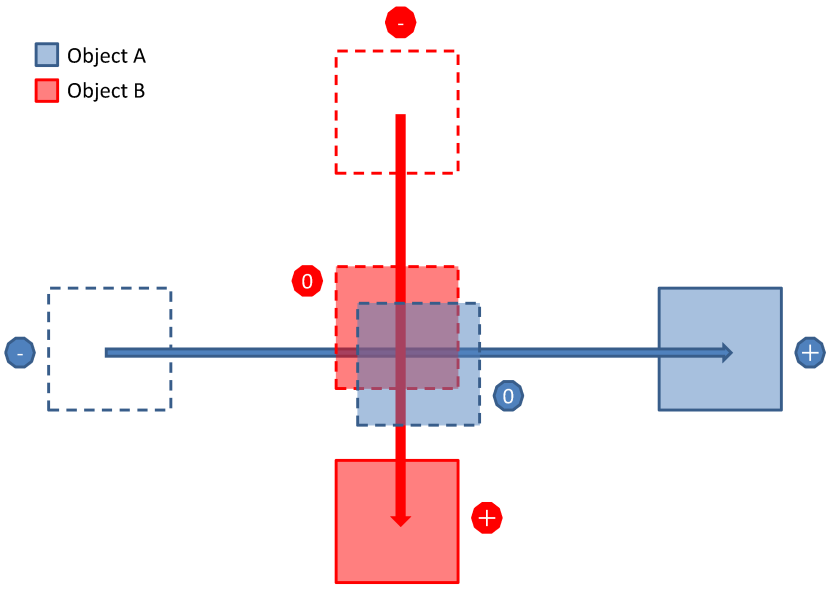


Figure 3.6: Interaction type ⓪ & ⓪ − (b).

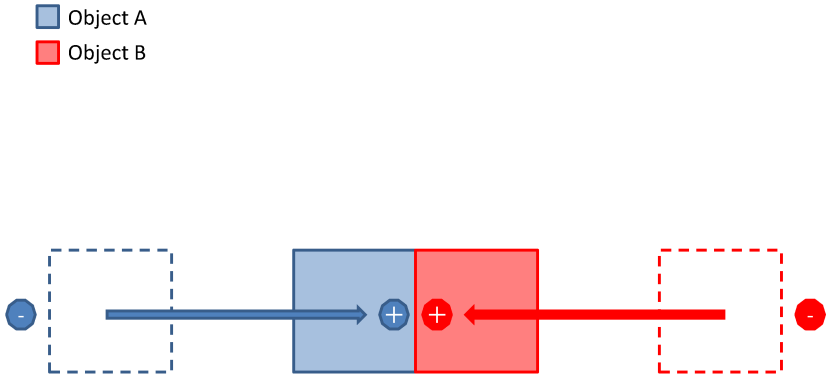


Figure 3.7: Interaction type ⊕ & ⊕.

⊕ & ⊕:

$$\begin{aligned} &\forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{end}}^A = t_{\text{end}}^B = T_{\text{end}}. \end{aligned} \tag{3.7}$$

⊖ & ⊖:

$$\begin{aligned} &\forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{start}}^A = t_{\text{start}}^B = T_{\text{start}}. \end{aligned} \tag{3.8}$$

⊕ & ⓪:

$$\begin{aligned} &\forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t_{\text{end}}^A = T_{\text{end}}. \end{aligned} \tag{3.9}$$

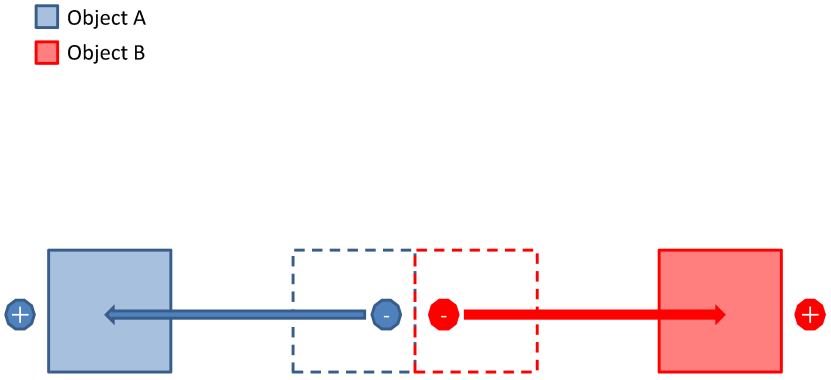


Figure 3.8: Interaction type ⊖ & ⊖.

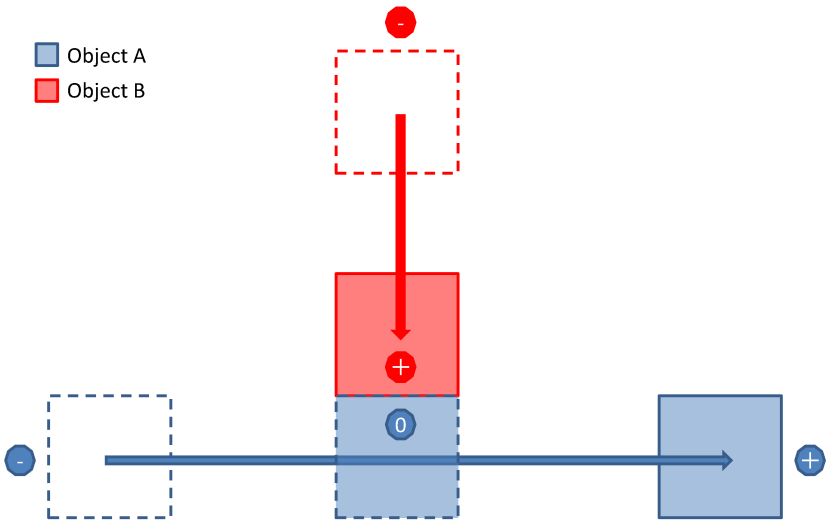


Figure 3.9: Interaction type ⊕ & ⓪.

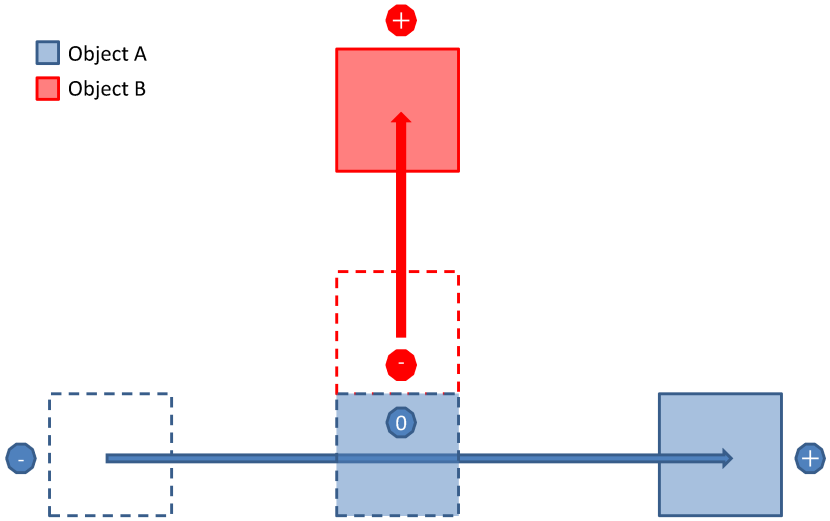


Figure 3.10: Interaction type ⊖ & ⓪.

⊖ & ⓪:

$$\begin{aligned} &\forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &t^A_{\text{start}} = T_{\text{start}}. \end{aligned} \tag{3.10}$$

Here $T_{\text{start}} = 0$ and $T_{\text{end}} = \tau$. $V_t'^A(x,y,z)$ represents a subset of $V_\tau^A(x,y,z)$ at time t. n and n' represent the set numbers of $(t_{\text{start}}, t_{\text{end}})$ from object A and B, respectively. $\forall v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B$ denotes all AHV voxels $v(x,y,z,t)$ whose 3D positions (x,y,z) have been occupied by both objects' motions for at least one moment during the time period τ.

Note that since types ⓪ & ⓪-(a), ⓪ & ⓪-(b) and ⓪ & ⓪-(c) do not contain any directional constraint, they can be easily applied onto AHVs composed of not only unidirectional actions. In that situation, the constraint equation for ⓪ & ⓪-(c) can be augmented as:

$$\begin{aligned} &\forall\, v_\tau^A \underset{(x,y,z)}{\cap} v_\tau^B, \\ &\bigcup_{i=1,i'=1}^{i=n,i'=n'} ([t^A_{\text{start}_i}, t^A_{\text{end}_i}] \cap [t^B_{\text{start}_{i'}}, t^B_{\text{end}_{i'}}]) = \phi. \end{aligned} \tag{3.11}$$

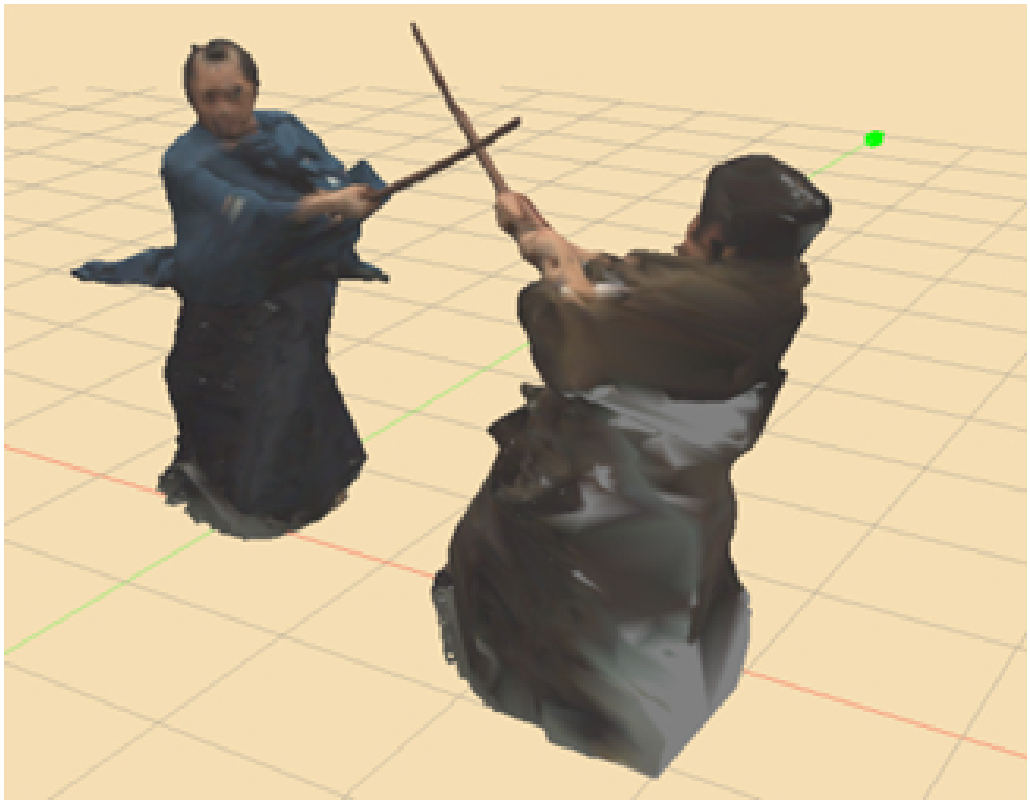


Figure 3.11: Attack and guard scene from sword fighting 3D video sequence.

As will be described in Chapter 5, all editing-related constraints for gaze actions belong to type ⓪ & ⓪-(c), so that with the above augmented constraint equation we can directly process the changeable gaze actions without decomposing them into unidirectional sub-actions.

This multi-party interaction dictionary provides us various spatiotemporal constraints to have objective criteria for performing the editing work. For example, the objects' body actions in an attack and guard scene (Fig. 3.11) from sword fighting sequence can be counted as type ⊕ & ⊕. Then on the interested part of the two AHVs modeling body actions, we require $t^A_{\text{end}} = t^B_{\text{end}} = T_{\text{end}}$, meaning that the two swords contact and only contact at the end of the whole action duration.

The multi-party interaction dictionary proposed in this Section enables to define three types of spatiotemporal constraints for future editing work, that are: (1) body action interacting type constraint, (2) mutual visibility constraint and (3) fidelity constraint. The body action interacting type constraint is mainly used in modeling the interaction event with contact, while the other two constraints can be applied to support the modeling of multi-party interaction event either with or without contact. The detailed definition and

application of them will be presented in Section 5.1.3.

3.3 Summary

In this chapter, we introduced the idea of Action History Volume and multi-party interaction dictionary. We first present the mathematical definition of the AHV. We also illustrate how to enable representing the spatiotemporal synchronization between multiple AHVs by assigning labels onto the AHV surface. A multi-party interaction dictionary is then proposed to model multi-party interaction events into different types of spatiotemporal synchronization between multiple AHVs.

This AHV-based multi-party interaction modeling method is used in the spatiotemporal editing work of 3D video, as will be presented in Chapter 5.

Chapter 4

3D Gaze Sensing for Gaze Action Modeling

Gaze action is an important communicative cue in multi-party interaction events. In order to use this cue in the 3D video editing work, methods need to be developed that perform 3D gaze estimation on 3D video data, and model their spatiotemporal structures. In this chapter we present our novel method that estimates the three-dimensional gaze direction (gaze vector) for 3D video data.

As described in Chapter 2, in the literature of accurate 3D gaze estimation from video, most conventional methods assume to have frontal face video of the object as their inputs [22]. Such methods are known to work robustly in practice [23], but they strictly limit the object's head motion in a very small range. Our 3D video editing scenario requires a gaze estimation method that allows the object (and his face) to move freely in the scene, and it should not rely on any kind of external gaze sensing equipment, which may disturb the object's natural behavior as well as his/her appearance. Therefore, we propose a "virtual frontal face video synthesis" approach that generates a frontal face video from regular multi-view videos. This approach realizes a non-contact and non-constrained gaze sensing and can be effectively used to perform gaze estimation on 3D video objects.

The ideas behind this method are as follows. Generally the accuracy of the reconstructed 3D shape data is limited due to errors in the calibration and shape reconstruction processes, which could mislead the gaze estimation and/or decrease its accuracy. Fortunately, the 3D face surface is rather flat, which allows many cameras to observe it, and moreover, it has symmetric properties in both 3D shape and surface texture. Thus a super-resolution technique with symmetry prior can be applied to increase the 3D shape accuracy and the image resolution, making full use of original multi-view images.

The overall processing scheme of the proposed gaze estimation method is as follows.

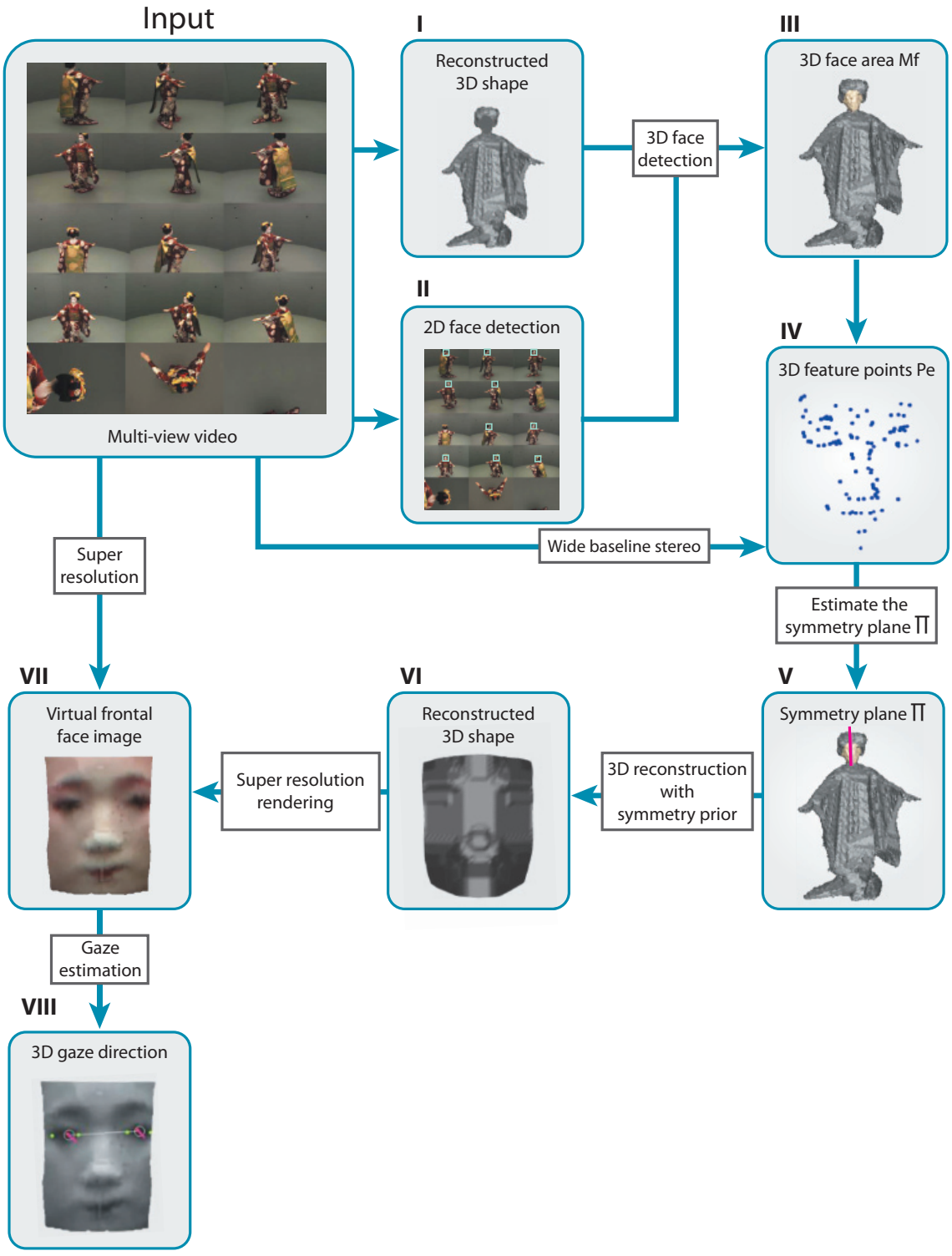


Figure 4.1: Computational processes for 3D gaze estimation. (*Copyright 2013 JSPS [80].*)

As is shown in Figure 4.1, given a sequence of 3D mesh data and corresponding multi-

view video data, we first extract 2D face regions in multi-view images to estimate a rough 3D face surface area in each 3D mesh (Figure 4.1 II and III). Then estimate the symmetry plane of the 3D face surface area by: (1) first extract 3D feature points in the estimated 3D face surface area and then, (2) generate the symmetry plane by evaluating symmetric properties among the feature points (Figure 4.1 IV and V). Next, we reconstruct an accurate and high resolution frontal face surface by applying a super-resolution 3D shape reconstruction technique with the symmetry prior (Figure 4.1 VI). Then a virtual frontal face image with super-resolution can be generated (Figure 4.1 VII). Finally, we estimate the 3D gaze from the virtual frontal face image using a 3D eyeball model (Figure 4.1 VIII).

For the following contents of this chapter, we first introduce our 3D face surface reconstruction method in Section 4.1, our virtual frontal face image synthesis algorithm in Section 4.2, and 3D gaze estimation algorithm in Section 4.3. We then evaluate our method with real data in Section 4.4. Finally, we summarize the proposed method in Section 4.6.

4.1 3D Face Surface Reconstruction using Symmetry Prior

In this section we present the super-resolution 3D shape reconstruction algorithm using symmetry prior from the 3D mesh and corresponding multi-view images. It consists of (1) 3D face area detection, (2) symmetry plane estimation and (3) 3D face surface reconstruction in super-resolution. The algorithm processes frames one-by-one sequentially.

4.1.1 3D face area detection

First we propose an algorithm to detect the 3D positions and directions of the object's face from multiple-view videos. The basic idea is to use a 3D mesh as a *voting space* for accumulating partial evidence produced by applying an ordinary 2D face detector to each of the multi-view images. The evidence accumulation enables to (1) eliminate false-positive face detections in 2D images and (2) localize an accurate 3D face area on the 3D mesh.

Let M denote a 3D mesh of an object and $I_i(i = 1, \ldots, N)$ a set of corresponding multi-view images captured by cameras $c_i(i = 1, \ldots, N)$. The face area detection algorithm (Figure 4.1 II and III) is defined as follows. Note that in what follows, Step X denotes the process X illustrated in Figure 4.1.

Figure 4.2: 2D face detection in multi-view images. Blue rectangles denote the detected 2D face candidate regions. (*Copyright 2013 JSPS [80].*)

First the algorithm detects a set of 2D face candidate regions F_i by applying a conventional 2D face detector to each I_i. The blue rectangles in Figure 4.2 show F_i for each image. It should be noted that F_i may include false-positive face areas due to texture patterns which accidentally look like a human face. Then all F_is are mapped onto M for evidence accumulation.

Step II Apply Viola-and-Jones face detector [36] to each image $I_i(i = 1, \ldots, N)$ to obtain a group of face candidate regions $F_i = \{f_{ij} | f_{ij} \in I_i, j = 1, \ldots, n_i\}$, where n_i denotes the number of face candidate regions in F_i.

Step III-1 Let $M = \{V, E\}$ denote a 3D mesh consisting of a vertex set V and an edge set E. For each vertex $v \in V$, compute a per-vertex "faceness" score $L(v)$ by the following method:

Step III-1-1 For each v, let $L(v) = 0$.

Step III-1-2 For each camera c_i, let v_i denote the projection of vertex v on image I_i. If v_i falls in F_i, then let $L(v) = L(v) + 1$.

Step III-2 Compute a set of vertices $V_L = \{v|L(v) > 0\}$, and partition it into disjoint subgroups of connected vertices $S = \{s_i|s_1 \cup s_2 \cup ...s_n = V_L, s_j \cap s_k = \emptyset\,(j \neq k)$, all vertices in s_i are connected. $\}$. Here n denotes the number of the subgroups.

Step III-3 For each vertex group s_i in S, calculate the average of $L(v)$ by:

$$\bar{L}_i(v) = \frac{\sum_{v \in s_i} L(v)}{N(s_i)}, \tag{4.1}$$

where $N(s_i)$ denotes the number of vertices in s_i.

Step III-4 Find the s_i with the largest $\bar{L}(v)$ and denote it by V_f.

$$V_f = \arg\max_{s_i} \bar{L}_i(v) \tag{4.2}$$

Return the sub-mesh $M_f = \{V_f, E_f\}$ as the 3D face area.

Figure 4.3 shows the detected 3D face area M_f.

4.1.2 Symmetry plane estimation

The assumption that human faces have symmetric properties in both 3D shape and surface texture allows us to reconstruct a more accurate 3D face surface than M_f and hence generate a higher resolution frontal face image than captured images.

The symmetry plane detection from M_f consists of two processes: (1) detect 3D feature points P_e on M_f (Figure 4.1 IV) and (2) apply RANSAC [38] to estimate the symmetry plane based on P_e (Figure 4.1 V).

3D feature points extraction

In order to find the symmetry plane that divides M_f into two symmetric parts, we first extract 3D feature points P_e on the local object surface specified by M_f. To avoid possible artifacts introduced by the texture generation, we apply a stereo-based edge feature

Figure 4.3: Detected 3D face area M_f painted in skin color. (*Copyright 2013 JSPS [80].*)

detection method to the multi-view images as illustrated in Figure 4.4. That is, we establish sparse but reliable 2D-to-2D correspondences to obtain 3D feature points by tri-

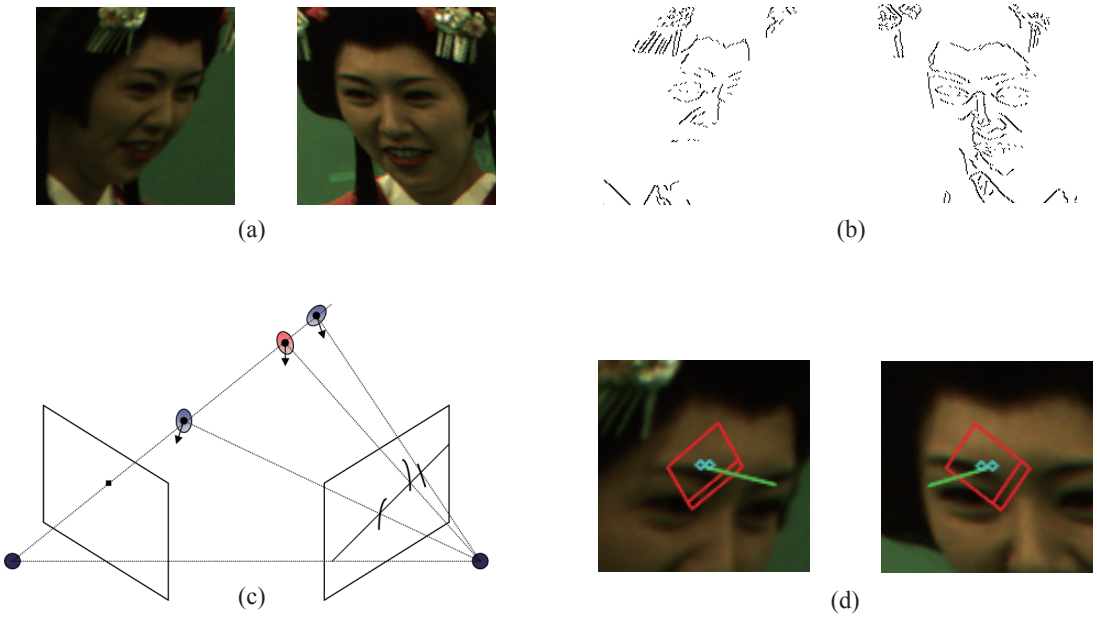


Figure 4.4: Matching based on edge features. (a) rectified images, (b) edge features crossing epipolar lines, (c) texture similarity computation with normal direction optimization, (d) an example of matched pair. In (d), the red rectangles illustrate the windows used to compute the texture similarity, the green lines the surface normals, and the blue circles the endpoints of the edge features. (*Copyright 2013 JSPS [80].*)

angulation [37]. This algorithm is based on the wide-baseline stereo by Furukawa [39] and augmented by a bi-directional uniqueness examination to improve the accuracy and robustness of the matching.

Step IV-1 Project M_f back onto the multi-view images to localize 2D face regions, respectively. Let c and c' denote a pair of cameras whose images include well captured 2D face regions. Rectify the images captured by c and c' for stereo matching (Figure 4.4(a)) and extract edge features from the 2D face regions in the rectified images.

Step IV-2 Eliminate edge features which do not cross the epipolar lines. Let I_E and I'_E denote the resultant edge feature images (Figure 4.4(b)). Let e denote a point on an edge feature in I_E, l' the corresponding epipolar line in I'_E, and $E' = \{e'_j | j = 1, \ldots, n\}$ the points on the edge features in I'_E intersecting with l'.

Step IV-3 Compute the texture similarity between e and $e'_j \in E'$ using the normal direction optimization [39] with the ZNCC photo consistency evaluation. (Figure 4.4(c)).

Let $\hat{e}'_j$ denote the point in E' which gives the best similarity. To enforce the uniqueness constraint, we accept the pair e and $\hat{e}'_j$ if and only if the similarity between them is significantly better than the second best pair. Otherwise we reject this pair and leave e without correspondence to avoid ambiguous matching.

Step IV-4 Validate the uniqueness of the correspondence in the opposite direction ($\hat{e}'_j \rightarrow e \in I_E$). If there is another edge feature point in I_E that has a comparable similarity value with $\hat{e}'_j$, reject this pair.

Step IV-5 By iterating the steps from IV-2 to IV-4 for all $e \in I_E$, we obtain the set of corresponding points between camera c and c'. We denote this set $P_{c,c'} = \{\langle p^i_c, p^i_{c'}\rangle \mid i = 1, \ldots, n_{c,c'}\}$, where $\langle p^i_c, p^i_{c'}\rangle$ denotes a corresponding point pair and $n_{c,c'}$ the number of obtained correspondences.

Step IV-6 By collecting $P_{c,c'}$ computed from all possible pairs of cameras that can observe the face area M_f, we can compute a set of 3D feature points, P_e, from a set of matching 2D point pairs.

Symmetry plane estimation using 3D feature points

Having computed the reliable 3D feature point set $P_e = \{p_i\}$ $(i = 1, \ldots, N)$ in Section 4.1.2, we then estimate the symmetry plane π from P_e as follows (Figure 4.1 V). The idea is to generate a candidate symmetry plane π and compare the texture pattern around p_i with that of its symmetric position with respect to π. If π is a valid symmetry plane, then the textures should be reasonably similar.

Step V-1 Randomly pick up two points p_i, p_j $(i \neq j) \in P_e$, and repeat the following processing for $K \leq N(N-1)/2$ times.

Step V-1-1 Compute the symmetry plane π_{ij} that makes p_i and p_j in the symmetric position.

Step V-1-2 Based on the hypothesized symmetry plane π_{ij} we can compute the symmetric position for each of the other $N-2$ points. Let $\breve{p}_k$ denote the symmetric position of p_k $(k \neq i, j)$. Then we compare the textures at p_k and $\breve{p}_k$. First we generate two $L \times L$ grids centered at p_k and $\breve{p}_k$ in the 3D space. Note that these two grids lie on the planes that are perpendicular to the hypothesized symmetric plane, and the distance between neighboring grid points is d, which

is a variable free to change according to the size of the 3D object. Since the 3D position of each grid point is computable, let p_k^{mn}, $\breve{p}_k^{mn}(0 \leq m \leq L, 0 \leq n \leq L)$ denote the grid points on the grids centered at p_k and $\breve{p}_k$. And let $Col(p_k^{mn})$ and $Col(\breve{p}_k^{mn})$ denote the RGB color vectors of the grid points p_k^{mn} and $\breve{p}_k^{mn}$ respectively, which are computed from the images by their best-observing cameras. Here we use M_f as the shape proxy for the state-based visibility evaluation[40]. Then the texture dissimilarity between p_k and $\breve{p}_k$, d_{p_k}, is computed as Sum-of-Absolute-Difference:

$$d_{p_k} = \sum_{0 \leq m \leq L, 0 \leq n \leq L} \left| Col(p_k^{mn}) - Col(\breve{p}_k^{mn}) \right|. \tag{4.3}$$

Note that if either p_k^{mn} or $\breve{p}_k^{mn}$ is located outside the estimated face area, the point pair is considered as an outlier and a fixed value $diff$ is set to $\left| Col(p_k^{mn}) - Col(\breve{p}_k^{mn}) \right|$. By computing d_{p_k} for all p_k $(k \neq i, j)$, we can evaluate the goodness of π_{ij} by

$$d_{ij} = \sum_{p_k \in P_e \setminus p_i, p_j} d_{p_k}. \tag{4.4}$$

Step V-2 Select the symmetry plane π_{ij} having the smallest $d_{i,j}$ as the symmetry plane π.

In experiments, we used $L = 4$ and $d = 5$ mm. The number of 3D feature points, N, was about a few hundreds, while changing from frame to frame.

4.1.3 3D shape reconstruction using symmetry prior

As is mentioned above, the shape reconstruction process in Figure 4.1 I estimates the 3D object surface geometry without any specific knowledge nor object model, which results in the limited reconstruction accuracy and the introduction of errors. By contrast, the 3D shape reconstruction algorithm in this section utilizes the knowledge of symmetric properties of the human face to attain more accurate and higher resolution 3D shape reconstruction (Figure 4.1 VI). The algorithm is similar to the mesh-deformation algorithm proposed by Nobuhara *et al.* [41], but employs the symmetry constraint in deforming the mesh. This section first describes how we model the 3D face surface by a mesh model, and then introduces how we can utilize the symmetry prior as a constraint on the mesh deformation.

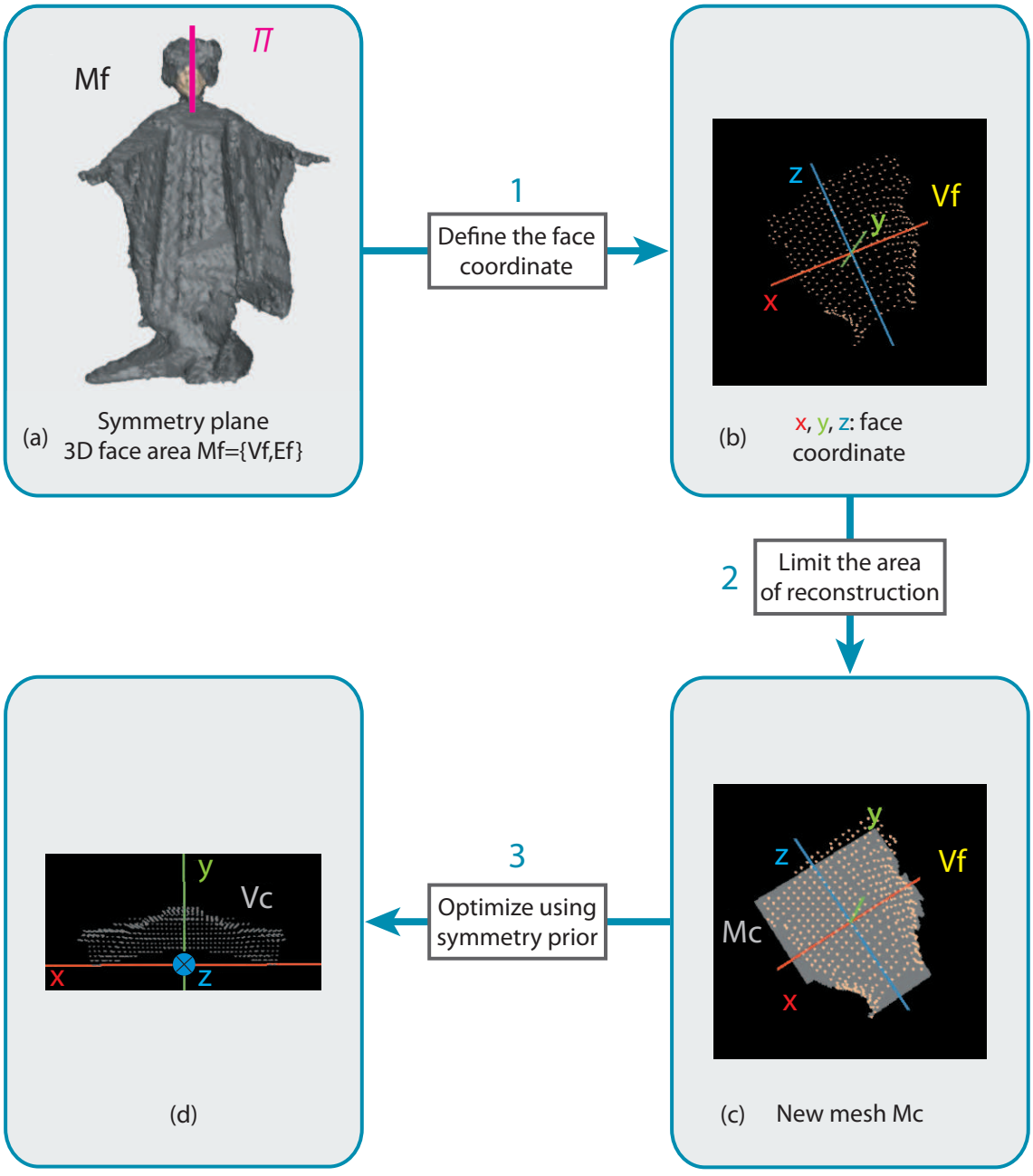


Figure 4.5: 3D face shape reconstruction using symmetry prior. (*Copyright 2013 JSPS [80].*)

The processes so far described have generated the 3D face area $M_f = \{V_f, E_f\}$ as a sub-area of the original 3D mesh surface M and estimated its symmetry plane π (Figure 4.5(a)). With this symmetry plane, we first define the 3D face coordinate system as illustrated in Figure 4.5(b): define the origin by the centroid of V_f and place the coordinate axes so that the symmetry plane π is aligned with $x = 0$ plane, the X-axis is defined by

the normal vector of π, and the Z-axis by the principal axis of the point distribution of V_f on π. The Y-axis is computed by the cross-product of the other axes.

Then we generate a new mesh $M_c = \{V_c, E_c\}$ to model the higher resolution 3D face surface: project M_f onto the $y = 0$ plane and define a bounded regular mesh M_c on the 2D projected region. The gray area in Figure 4.5(c) illustrates M_c. That is, V_c and E_c denote the set of grid points and edges in this projected region, respectively. Note that the sampling pitch by the regular grid can be designed to increase the spatial resolution.

With this modeling, the 3D face surface reconstruction problem is transformed to that of finding the appropriate y value of each regular grid point in V_c (Figure 4.5(d)). Here the technical problems to be solved are (1) how we can introduce the symmetry constraint into the mesh deformation and (2) how we can find the optimal y values for V_c.

First, we represent the symmetry prior by

$$y = f(x,z) = f(-x,z), \tag{4.5}$$

where the function $f(x,z)$ returns the y value of the grid point at (x,z). Then, introduce the following discrete representation of y values:

$$y = \alpha i, \tag{4.6}$$

where i denotes an integer within a certain range, and α specifies the resolution of possible y values.

This discrete modeling allows us to formalize the shape reconstruction problem as a multi-labeling problem. That is, we can formulate the shape reconstruction with the symmetry prior as the minimization of the following objective function:

$$\mathcal{E}(M_c) = \sum_{v \in V_c, v_x \geq 0} \mathcal{E}_p(i_v) + \sum_{(u,v) \in E_c, u_x, v_x \geq 0} \mathcal{E}_c(i_u, i_v), \tag{4.7}$$

where v_x and u_x denote the x coordinate values of v and $u \in V_c$ respectively, and i_v and i_u integer labels to specify y values at v and u respectively. $\mathcal{E}_p(i_v)$ denotes the photo-consistency evaluation function at v and its symmetric position $\breve{v}$. That is,

$$\begin{aligned} \mathcal{E}_p(i_v) &= \rho(v_x, \alpha i_v, v_z) + \rho(\breve{v}_x, \alpha i_{\breve{v}}, \breve{v}_z) \\ &= \rho(v_x, \alpha i_v, v_z) + \rho(-v_x, \alpha i_v, v_z), \end{aligned} \tag{4.8}$$

where $\rho()$ denotes the photo-consistency evaluation function based on the state-based visibility with M as the shape proxy[40]. $\mathcal{E}_c(i_u, i_v)$ evaluates the smoothness in the y

direction between a pair of connected grid points v and u:

$$\mathcal{E}_c(i_u, i_v) = \kappa|\alpha i_u - \alpha i_v| \tag{4.9}$$

where κ is a weighting factor to balance the photo-consistency and smoothness terms. This formalization forces the mesh deformation to satisfy the symmetry constraint defined by Equation (4.5). We solve this minimization problem by belief-propagation[42], and obtain the 3D face surface satisfying both the photo-consistency and the symmetry constraint simultaneously.

In experiments, we used $\kappa = 1.0$, $\alpha = 1$ mm, and 2.5 mm grid resolution for M_c. Note that the original 3D mesh resolution, *i.e.* the average distance between adjacent vertices of M_f was about 4.7 mm.

4.2 Virtual Frontal Face Image Synthesis

Generally, super-resolution techniques for estimated 3D shape from multi-view videos can be categorized into two groups: view-independent and view-dependent. The first group generates a super-resolution texture of the 3D surface[28]. The generated texture is optimized over the object surface, and suits for view-independent rendering. The second group generates a super-resolution rendering of the object for a specified viewpoint[29]. The generated image is optimized for the viewpoint, and therefore suits well for our virtual frontal face image synthesis. Based on this understanding, here we employ the view-dependent approach which can generate an optimized super-resolution image of the object 3D face.

With the optimized M_c, the virtual frontal view of M_c is generated for gaze estimation (Figure 4.1 VII): (1) locate a virtual camera with focal length f at $(0, P_{cam}, 0)$ and align its view direction at $(0, 0, 0)$ in the 3D face coordinate system defined in Section 4.1.3, and then (2) generate the virtual frontal face image by rendering M_c from the virtual camera by the super-resolution technique proposed by Tung *et al.* [29]:

Step VII-1 Set a high-resolution pixel grid on the image plane of the virtual camera.

Step VII-2 Project each pixel of the original multi-view images, say source pixels, onto the pixel grid via M_c. That is, back project the source pixels onto M_c first, and then project the points on M_c to the pixel grid of the virtual camera. In this process we choose the nearest grid point as the final projection point of

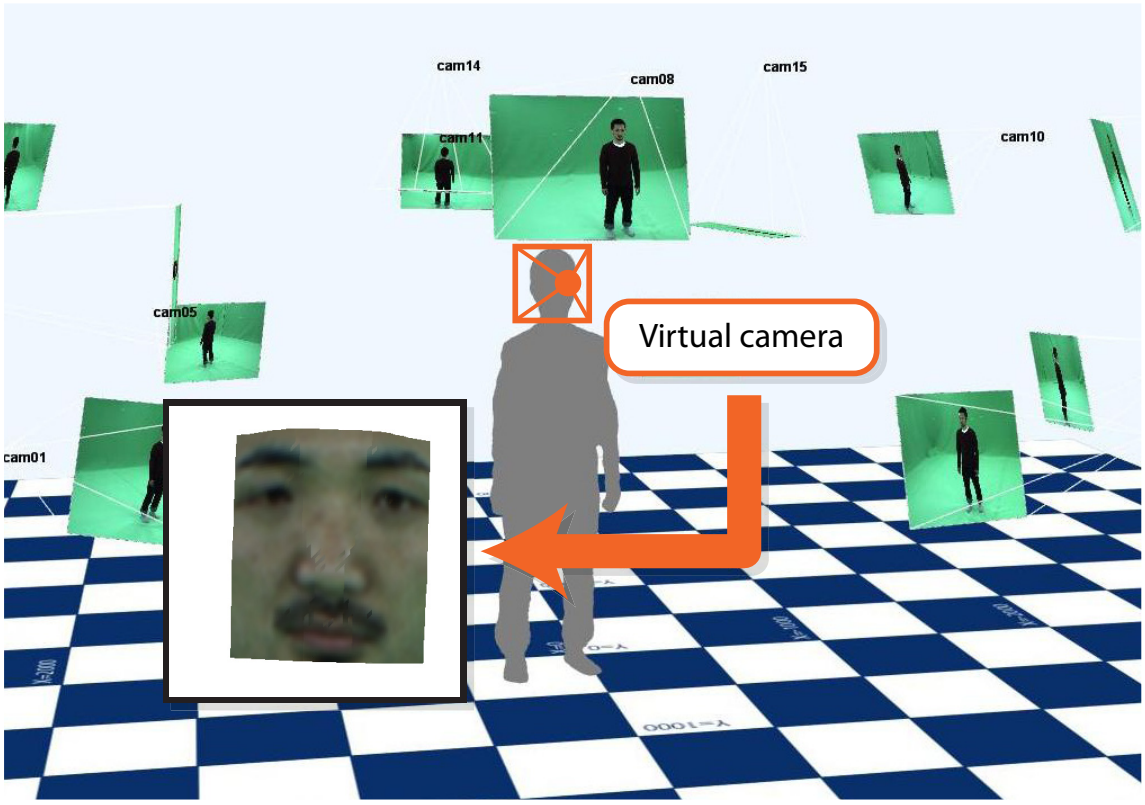


Figure 4.6: Virtual frontal face image synthesis. (*Copyright 2013 JSPS [80].*)

each source pixel. In addition we ignore source pixels if their projections are occluded by M.

Step VII-3 For each grid point with source pixel projections, compute its color by averaging associated source pixel colors. Otherwise, interpolate the grid point color using colors of its neighbors.

Figure 4.6 shows a synthesized virtual front face image, where the image resolution is increased by the super-resolution rendering process.

In experiments, we used $f = 430\,\text{mm}$, $P_{cam} = 500\,\text{mm}$, and the virtual face image plane of $160\,\text{mm} \times 160\,\text{mm}$ sampled with 400×400 pixels. Considering that the average of y values in a 3D face area is about 15 mm, the size of the virtual image pixel projected on the 3D face surface is about $0.45\,\text{mm} \times 0.45\,\text{mm}$, which is much higher than 4.7 mm, the average distance between adjacent vertices in the original 3D mesh. Note that the resolution of the original multi-view images is higher than that of the original 3D mesh. It was estimated about at most $1.5mm$ on the face area. That is, the super-resolution attained about three times higher resolution than the original images.

4.3 Gaze Estimation using 3D Eyeball Model

At the last stage, we propose to introduce a 3D eyeball model to estimate the object's gaze direction (Figure 4.1 VIII). Figure 4.7 illustrates the structure of the model. The red arrow indicates the 3D gaze direction, and θ and φ denote the horizontal and vertical rotation angles of the eyeball, respectively. This model is designed based on the following three assumptions:

1. The eyeball is fixed inside the eye socket and it can rotate horizontally and vertically around the eyeball center.

2. The gaze direction is defined by the 3D vector pointing from the eyeball center to the iris center.

3. The radius of the eyeball is equal to the diameter of the iris. This assumption is made based on medical statics data.

To apply this model to the 3D gaze estimation, the eyeball model of the object should be estimated first by the following off-line process:

Step VIII-1 Collect virtual frontal face images in which eyes look straight forward by hand.

Step VIII-2 For each image, detect the following eye feature points (Figure 4.8) for each eye: 2D eye corners, q_a and q_e, 2D iris center, q_c, and the intersecting points between the iris border and the eye corner line connecting q_a and q_e, q_b and q_d. The eye corners are located by the AAM [31], and the iris is detected by applying Kawaguchi *et al.* [43]'s method. Note that all feature points for the right eye illustrated in Figure 4.8 are mirrored with respect to the symmetry plane to represent those for the left eye.

Step VIII-3 For each eye, let d denote the average 3D diameter of the iris, and consequently the eyeball radius. The 3D diameter of the iris is defined by the 3D distance between p_b and p_d on the face surface M_c, which are obtained by back-projecting q_b and q_d onto M_c respectively.

Step VIII-4 For each eye, compute the average 2D relative position t of the iris center q_c with respect to the eye corners q_a and q_e. That is, t denotes the weighting parameter to represent q_c by the weighted average of q_a and q_e: $q_c = (1 - t)q_a + tq_e$ where $t = |q_c - q_a| \,/\, |q_e - q_a|$.

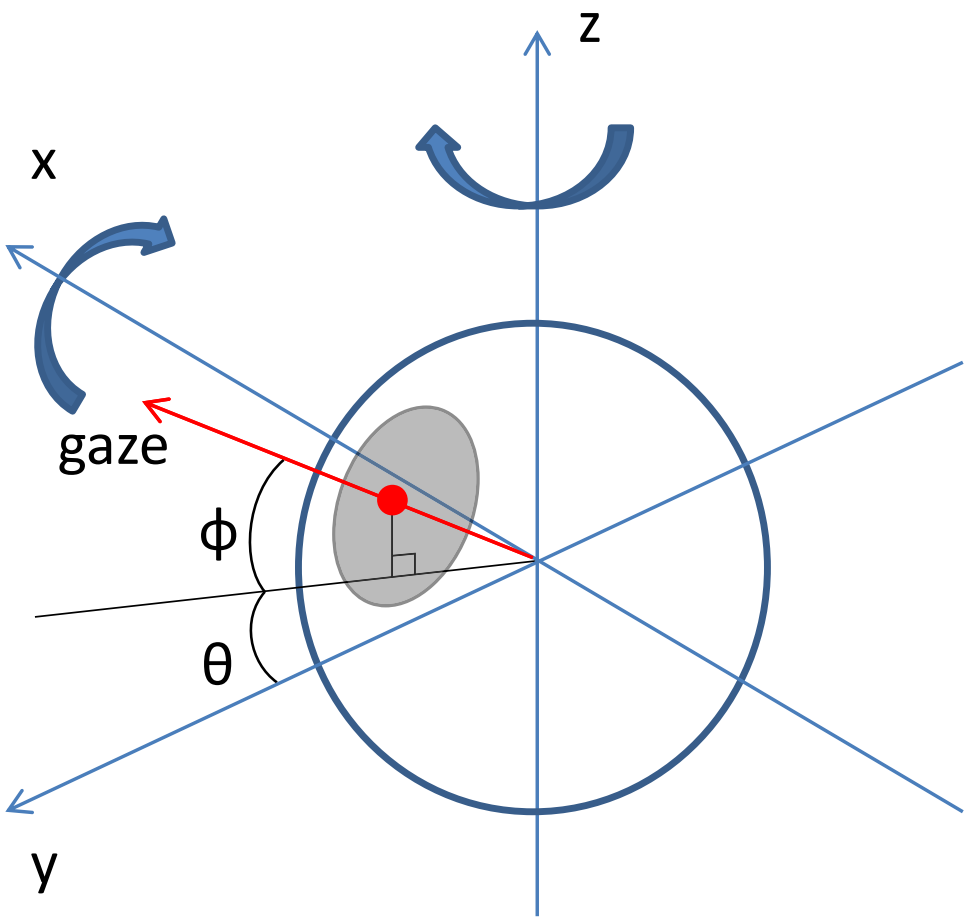


Figure 4.7: 3D eyeball model. (*Copyright 2013 JSPS [80].*)

This process estimates the eye model parameters for the left and right eyes respectively: d_{left} and t_{left}, and d_{right} and t_{right}. In what follows, we eliminate the suffix for simplicity.

With the eye ball model parameters d and t, compute the 3D gaze directions of the left and right eyes from each 3D video frame by the following process (Figures 4.8 and 4.9). Note that all 3D points as well as the virtual frontal face image in the 3D gaze estimation below are represented in the face coordinated system defined in Section 4.1.3 and Figure 4.5(b), which is dynamically defined depending on the 3D face position and direction in each 3D video frame.

Step VIII-5 Apply the following process to the left and right eyes, respectively.

Step VIII-6 Detect 2D eye corners q_a and q_e, and the iris center q_c from the synthesized

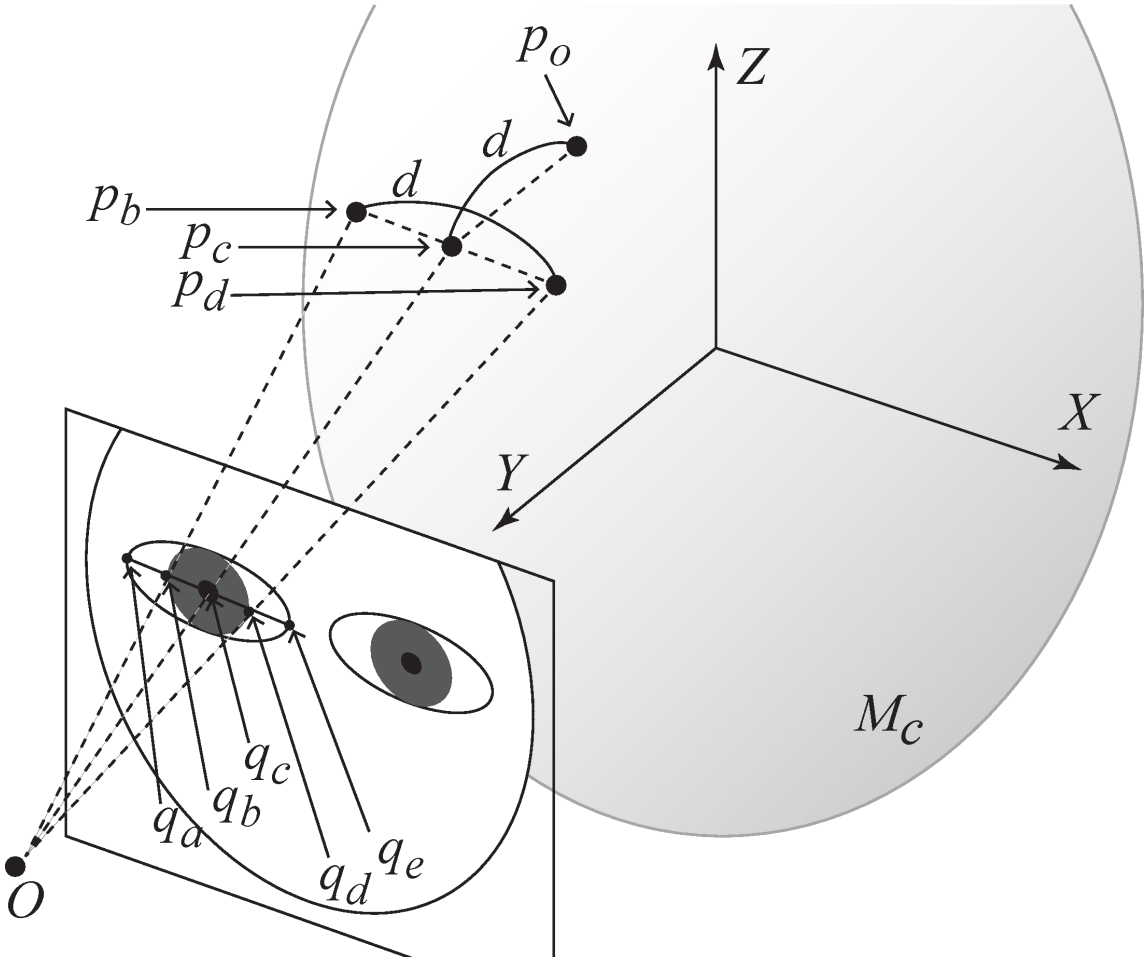


Figure 4.8: Eyeball center position estimation. (*Copyright 2013 JSPS [80].*)

frontal face image.

Step VIII-7 Compute the 3D iris center position p_c by back-projecting q_c onto M_c.

Step VIII-8 Compute the 3D eyeball center p_o by

$$p_o = p_{\tilde{c}} + (0, -d, 0)^\top. \tag{4.10}$$

Here $p_{\tilde{c}}$ denotes the back-projection of $q_{\tilde{c}} = (1-t)q_a + tq_e$ onto M_c, where $q_{\tilde{c}}$ represents the 2D position of the assumed iris center if the eye were looking straight forward.

Step VIII-9 Finally the 3D gaze direction is given as the line passing through p_o and p_c.

It should be noted that here the gaze vectors are estimated for the left eye and right eye, respectively. While to keep the simplicity of gaze action modeling, the average of

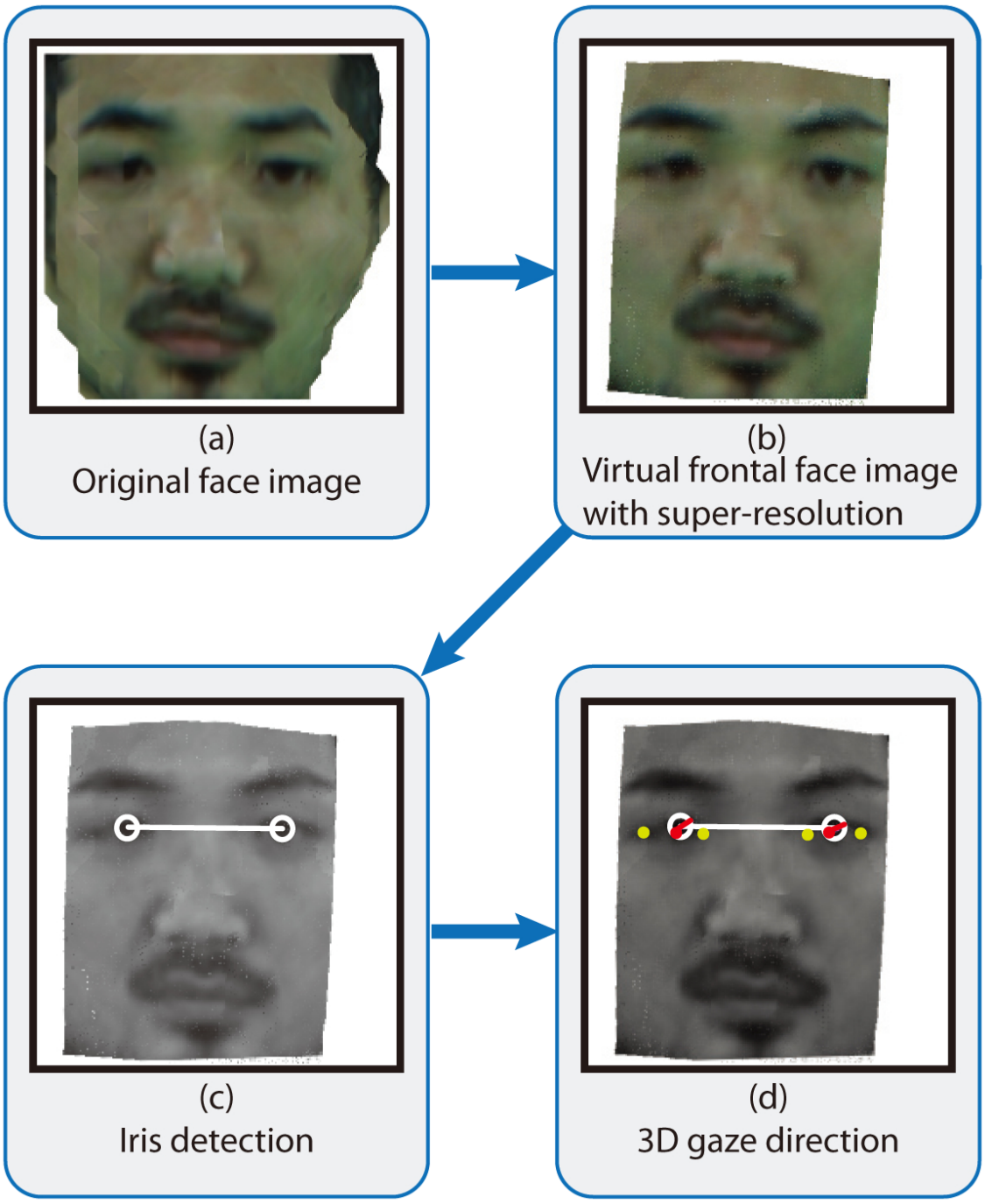


Figure 4.9: Gaze estimation process. (a) original face image generated based on the original 3D mesh M (b) virtual super-resolution frontal face image generated based on the reconstructed face surface M_c (d) detected irises (circles) (e) estimated eye corners (green dots) and gaze directions (red lines). (*Copyright 2013 JSPS [80].*)

the gaze vectors estimated from the object's two eyes, which is denoted as $\overrightarrow{G_i}$, is used to

generate the gazing cone as described in Section 3.1.3. The 3D position of the center of the two eyes can also be easily computed since the 3D eyeball center p_o of each eye is already computed in Step VIII-8.

4.4 Performance Evaluation

In this section we evaluate the performance of the proposed method with real data.

4.4.1 Shape reconstruction using symmetry prior

First, we present the analysis on how the accuracy of the reconstructed 3D face shape is improved by introducing the symmetry prior.

Experiment setup

For this evaluation we used two sets of data in doing the experiment. Data A (Figure 4.10) is captured by 15 calibrated UXGA cameras running at 25 HZ with 1 msec shutter speed, while Data B (Figure 4.11) is captured by 16 calibrated UXGA cameras running at 25 Hz with 1 msec shutter speed.

Evaluation method

We measure the contribution of the symmetry prior to the reconstruction accuracy by means of leave-one-out experiments. We keep one camera c_f for evaluation, and use the other 15 cameras to render the face image viewed from camera c_f by the rendering algorithm described in Section 4.2. Note that the size and resolution of the rendered image is adjusted to coincide with that of the image captured by c_f. Let I'_f denote the rendered image. Then we compute the mean-squared-error between the rendered image I'_f and the originally captured image I_f:

$$\text{MSE} = \frac{1}{N} \sum_{(x,y) \in I_f} (I_f(x,y) - I'_f(x,y))^2, \tag{4.11}$$

where N is the total number of effective pixels in I'_f.

In this experiment we compute I'_f in two ways as follows: (1) the virtual view image with the original 3D shape. (2) the virtual view image with the reconstructed 3D shape using symmetry prior. By comparing these two types of I'_f with the original image I separately, we can comprehensively evaluate the effect of introducing the symmetry prior.

Figure 4.10: Input multi-view data A. (*Copyright 2013 JSPS [80].*)

Experiment results

Figure 4.12 and 4.13 illustrate the difference images between the original captured image and the synthesized virtual frontal face image, generated with the original 3D shape and

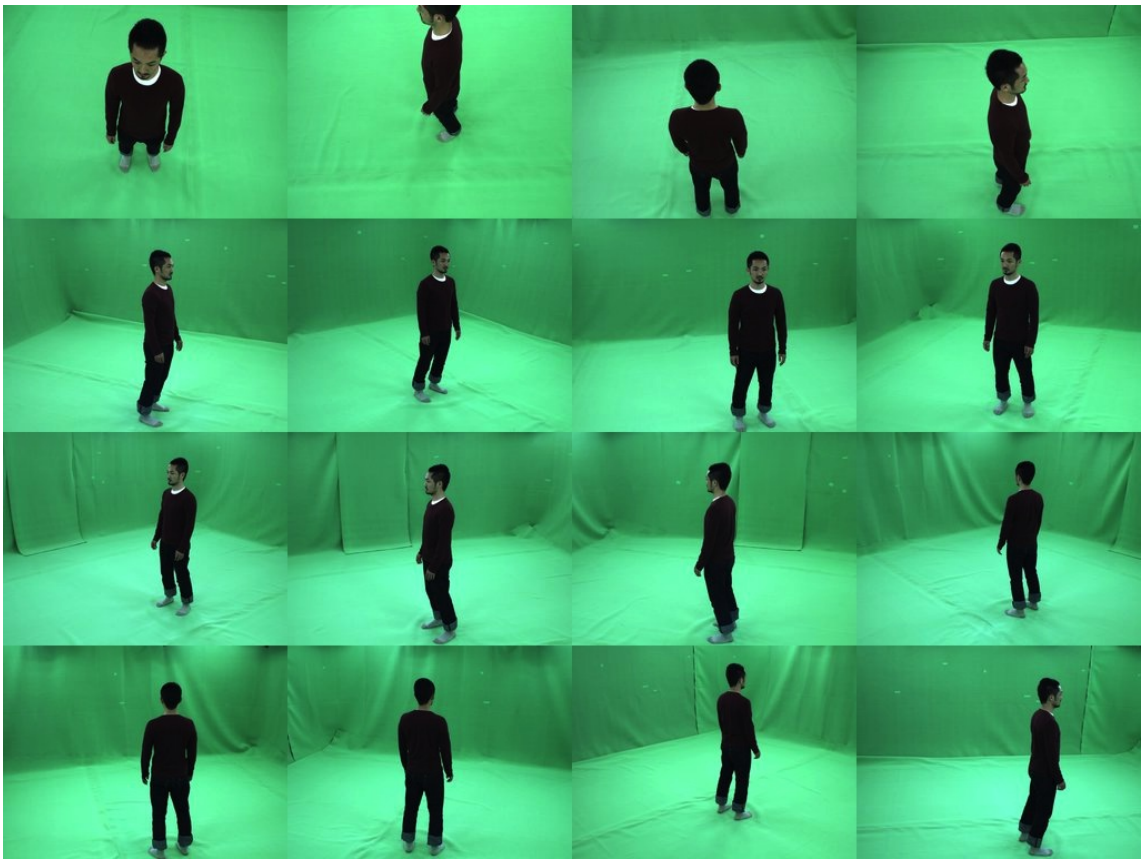


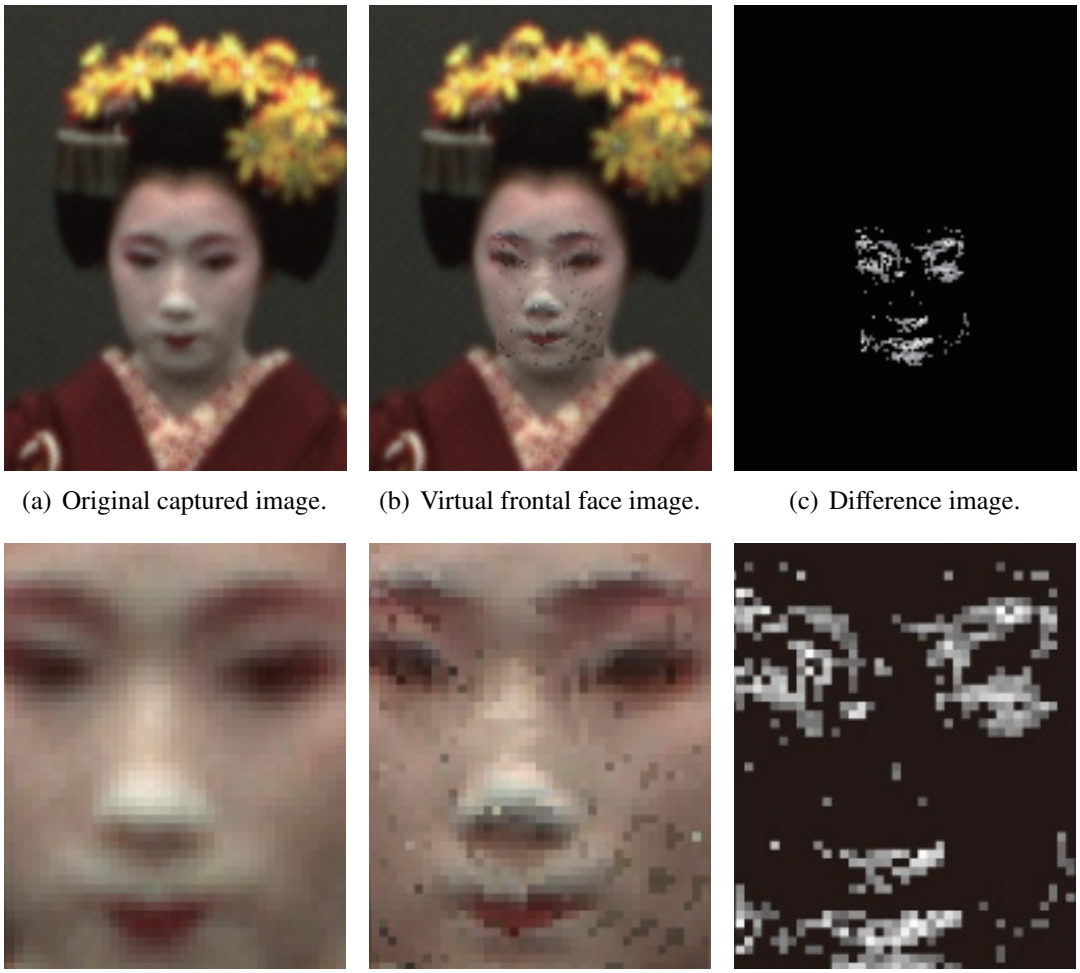
Figure 4.11: Input multi-view data B. (*Copyright 2013 JSPS [80].*)

Table 4.1: MSE with the original captured image.

	MSE	effective pixels
proposed method	10.23	1900
original 3D shape	11.07	2223

the reconstructed 3D shape using symmetry prior, respectively. In Figure 4.12, the synthesized face image in the middle is less blurred than the original captured image, and the differences mainly occur on the edges, proving that the synthesized virtual frontal face image with the proposed method has higher resolution than the original captured image. As for the one with the original 3D face shape, obvious differences appear on the cheeks as well as the edges. Table 4.1 illustrates that the mean-squared-error MSE of the proposed method is smaller than using the original 3D face shape.

Figure 4.14 shows the mean-squared-errors over 100 continuously captured frames using Data B. It can be observed that the symmetry prior contributes to improve the fidelity



(a) Original captured image. (b) Virtual frontal face image. (c) Difference image.

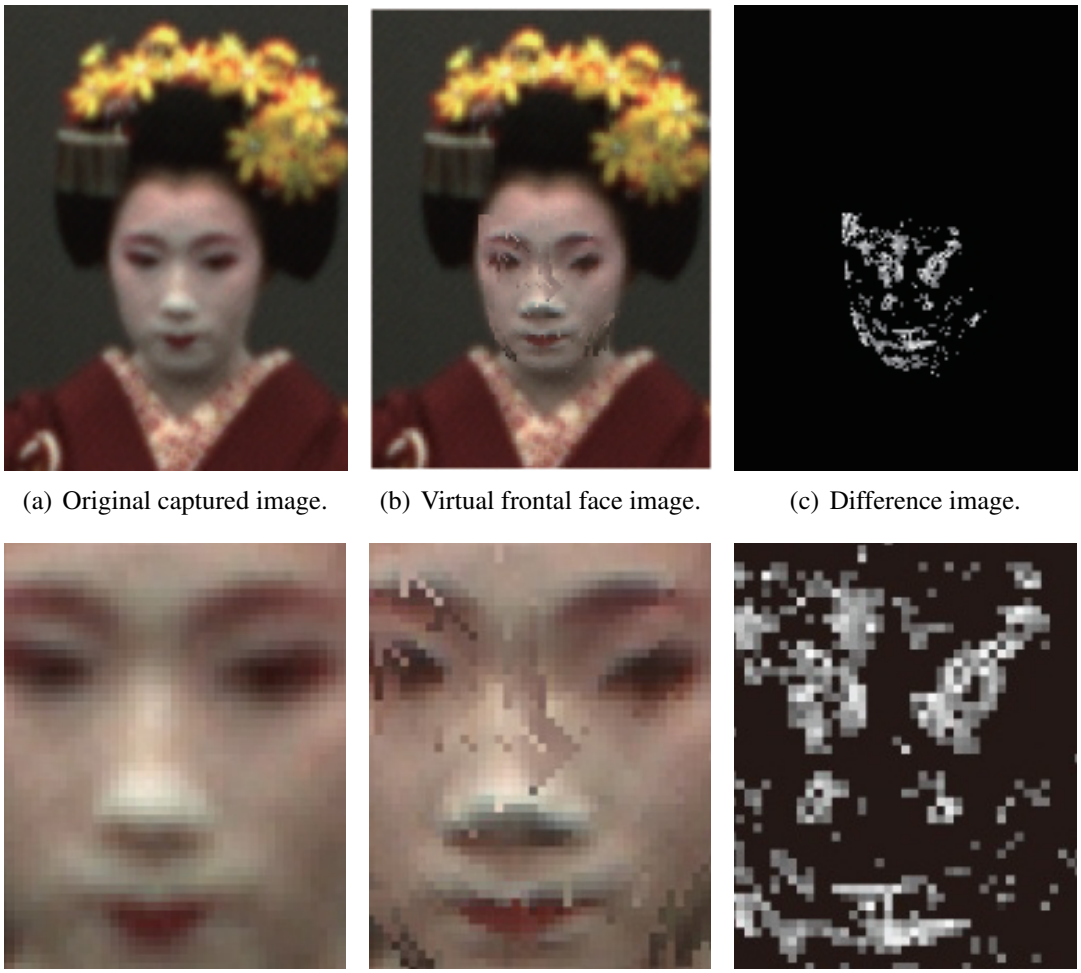
(d) Enlarged Original captured image. (e) Enlarged Virtual frontal face image. (f) Enlarged Difference image.

Figure 4.12: Difference image with the proposed method. (*Copyright 2013 JSPS [80].*)

of rendering and hence improve the reconstruction accuracy.

4.4.2 Gaze estimation

To prove the effectiveness of our gaze estimation method, we prepared the input multi-view videos captured with three different people for evaluation (Figure 4.15). We have also conducted the experiment with an effective commercial gaze tracking device Tobii X120 Gaze Tracker in the same environment for comparison.



(a) Original captured image. (b) Virtual frontal face image. (c) Difference image.

(d) Enlarged Original captured image. (e) Enlarged Virtual frontal face image. (f) Enlarged Difference image.

Figure 4.13: Difference image with the original 3D shape. (*Copyright 2013 JSPS [80].*)

Experiment setup

In order to quantitatively evaluate the accuracy of the gaze estimation processing, we designed our experiment as follows. Figure 4.16 illustrates the experimental environments. A human subject stands at about 2.5 m away from the wall and looks at (1) horizontally aligned markers one by one, and (2) vertically aligned markers one by one (Figure 4.16 left and right respectively). For each marker, its 3D position p_m in the object oriented coordinate system is measured manually. Figure 4.17 is the template used for iris detection based on Kawaguchi *et al.* [43]'s method, with the size of 300×122 pixels.

As is shown in Figure 4.18, experiments using a Tobii X120 Eye Tracker are conducted as well in the same environment. The object stands at the same position as in the experiment with multi-view cameras. A Tobii X120 Eye Tracker is set in front of

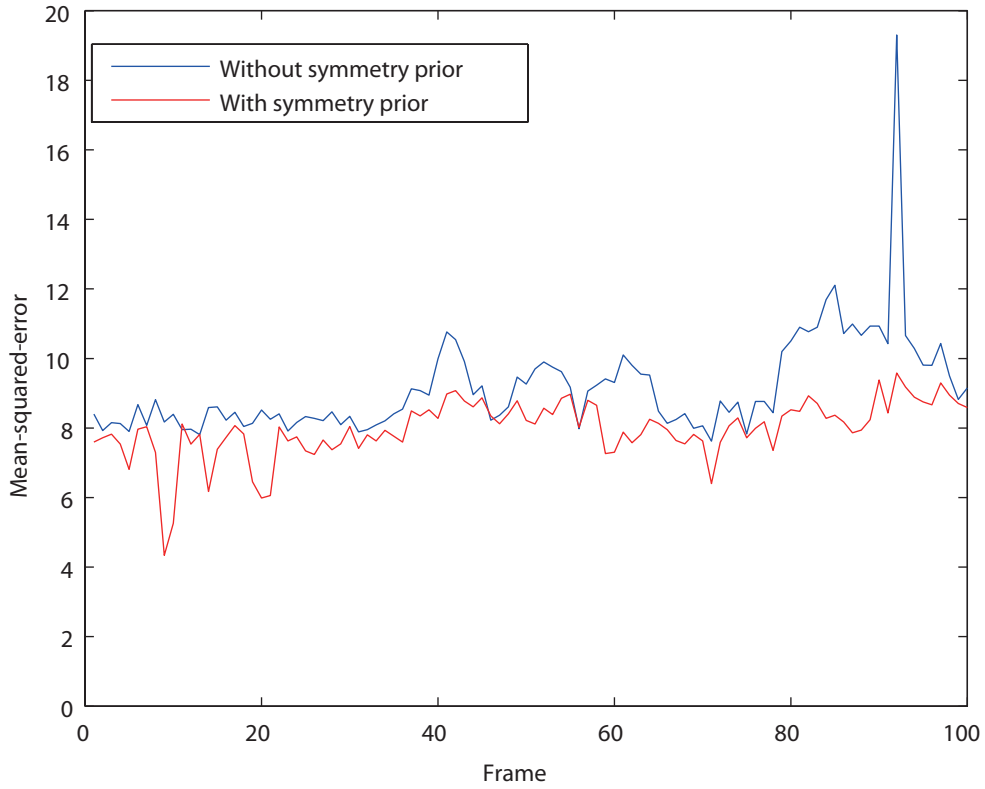


Figure 4.14: Mean-squared-errors between the synthesized and the original images. (*Copyright 2013 JSPS [80].*)

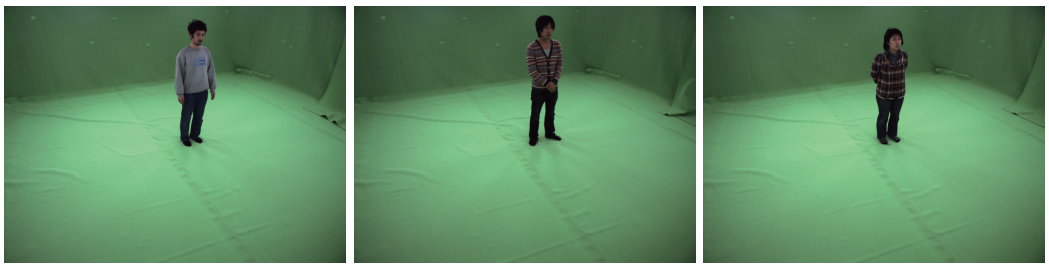


Figure 4.15: Multi-view input data with three different objects. (*Copyright 2013 JSPS [80].*)

the object for iris and gaze tracking, and a projector is used to project the marker image onto the wall for the object to look at. Since the positions of the two outer markers in the horizontal direction (Figure4.16, left) are out of the effective region of Tobii, we only project the other five markers onto the wall. It should be noted that the marker image is well designed to project each marker into the same position as the corresponding one's

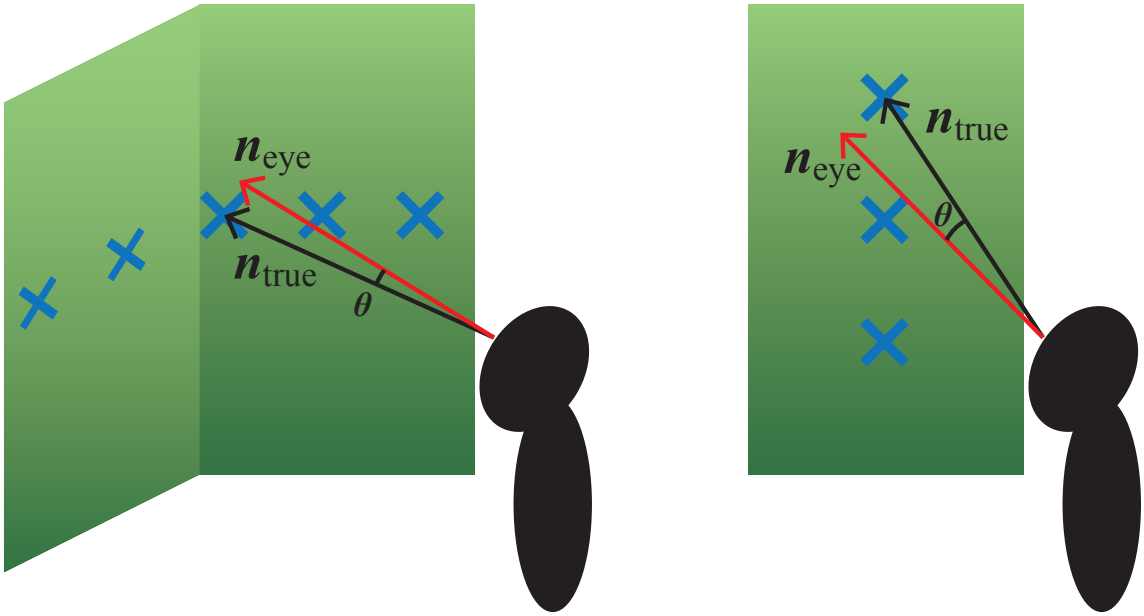


Figure 4.16: Gaze estimation error evaluation. (*Copyright 2013 JSPS [80].*)

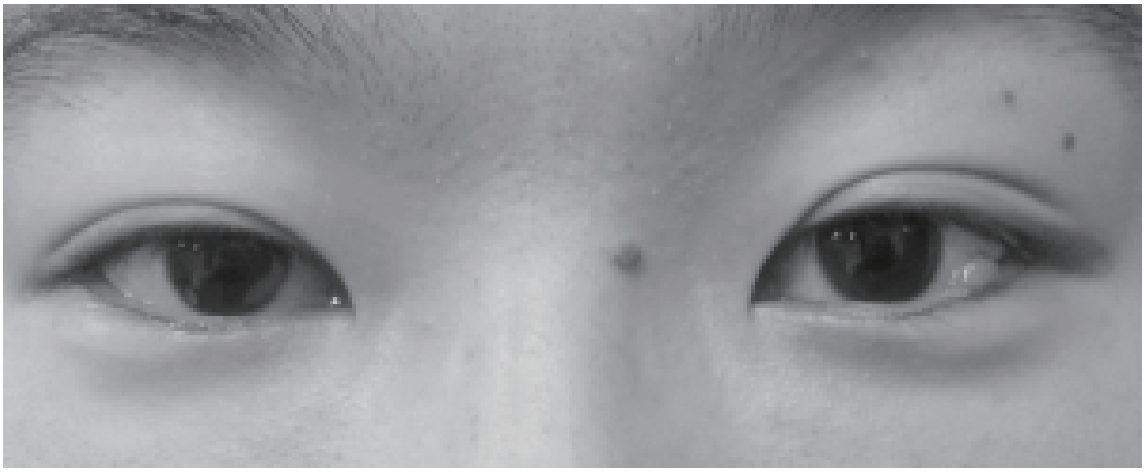


Figure 4.17: Template for iris detection. (*Copyright 2013 JSPS [80].*)

positron in the multi-view camera experiment. Besides, we manually set two markers onto the positions of the two outer ones in the horizontal direction (one of them is highlighted with a red circle in Figure 4.18), which could be used to estimate the performance of Tobii when the gazing position is outside its effective region. Table 4.2 illustrates the technical specifications of the Tobii X120 Eye Tracker being used in our experiment.

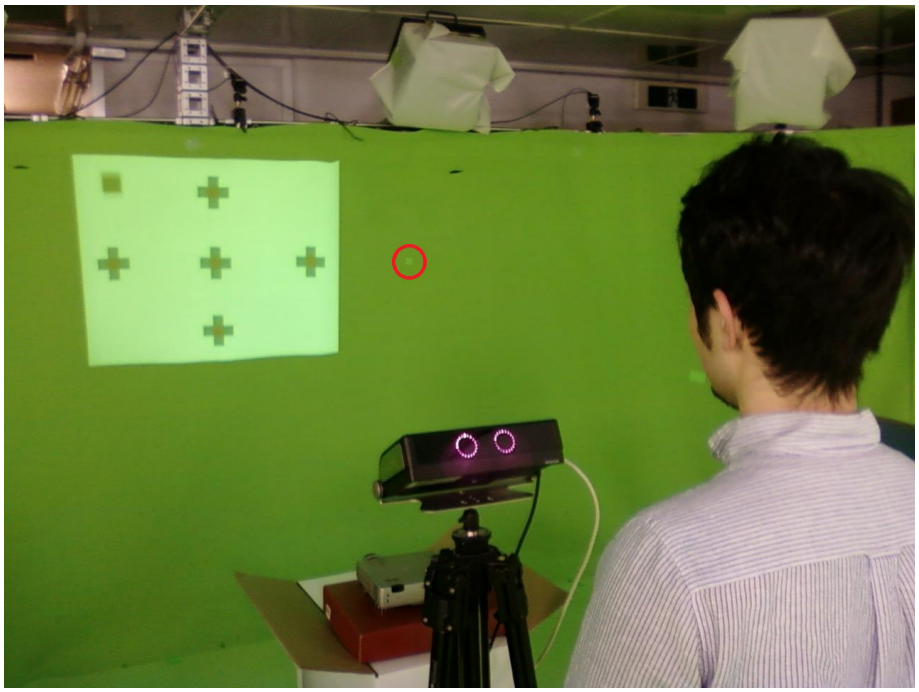


Figure 4.18: Experiment with Tobii X120 Eye Tracker. (*Copyright 2013 JSPS [80].*)

Table 4.2: Tobii X120 technical specifications.

Data rate	60 HZ
Latency	30-35 ms
Time to tracking recovery	Average 100ms
Max gaze angles	35 degrees
Freedom of Head Movement	44*22*30 cm at 70 cm (Width*Height*Depth)
Tracker field of view	36*22*30 cm at 70 cm (Width*Height*Depth)
Top head-motion speed	35 cm/second
Eye tracking technique	Both bright and dark pupil tracking

Evaluation method

In the multi-view video data, we first selected those video frames where the subject was stably looking at each marker. Then, for each selected video frame, apply the above mentioned gaze estimation processes from Step I to Step VIII to obtain the estimated 3D gazing vector $\boldsymbol{n}_{\text{eye}}$. Note that the gaze estimation is conducted for the left and right eyes independently, meaning that we have $\boldsymbol{n}_{\text{eye}}$ for each eye. The ground-truth 3D gazing direction vector $\boldsymbol{n}_{\text{true}}$ is defined by a 3D vector from the eye ball center p_o computed by Equation (4.10) to each 3D marker position. Then, the angular error of the 3D gaze

estimation in each selected video frame is computed for each eye by

$$\theta = \arccos\left(\frac{\boldsymbol{n}_{\text{eye}} \cdot \boldsymbol{n}_{\text{true}}}{|\boldsymbol{n}_{\text{eye}}||\boldsymbol{n}_{\text{true}}|}\right) \tag{4.12}$$

On the other hand, a calibrated Tobii X120 Eye Tracker continuously outputs the estimated gazing position on the wall. Since Tobii is not designed to track the gaze of the two eyes separately, we compute the estimated 3D gazing direction vector $\boldsymbol{n}_{\text{eye}}$ as one 3D vector from the middle position of the two eye ball centers to each estimated marker position. Then the angular error of the Tobii's gaze estimation in each frame can be computed by Equation (4.12).

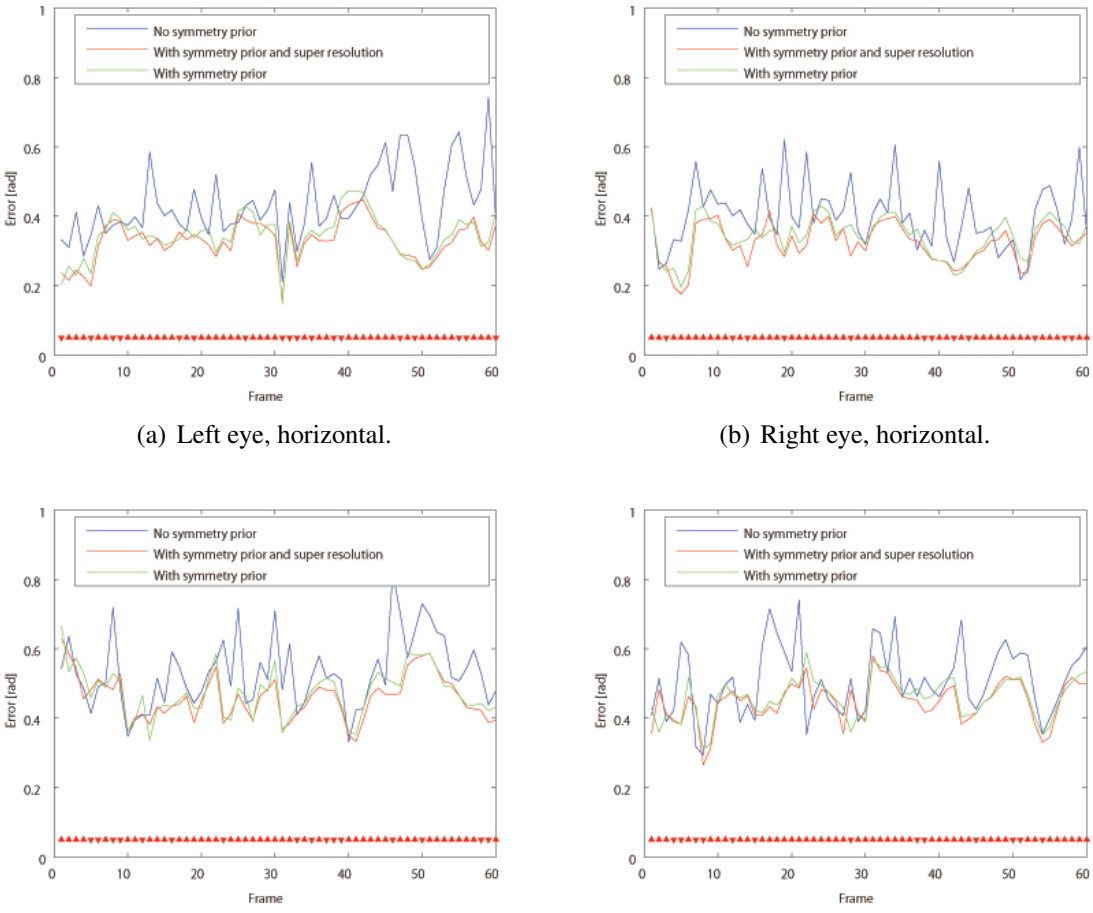
Experiment results

In the analysis of the proposed method, the angular gaze estimation errors are evaluated for the left and right eyes as well as for the horizontal and vertical directions, respectively, which gives four different error evaluation results as shown in Figures 4.19(a), 4.19(b), 4.19(c) and 4.19(d).

In each figure, three computational methods are compared: without the symmetry prior, with the symmetry prior alone, and with both the symmetry prior and the super-resolution image rendering technique. The horizontal axis in each figure denotes the selected frame IDs where the subject was stably looking at each marker. The upward and downward triangles at the bottom in each figure denote the signs (*i.e.* positive or negative) of the errors by the method with both the symmetry prior and the super-resolution image rendering technique. Table 4.3 shows the average errors for the first and the third methods.

Table 4.4 compares the numbers of iris detection failures in 100 continuously captured frames. In all results, the symmetry prior improved the stability of the iris detection and the accuracy of the gazing direction estimation, while the improvement by the super resolution is limited. This is because the performance of the iris localization is not so accurate. As is well known, errors in the horizontal direction are much smaller than those in the vertical direction, because of the shape and movable range of human eyes. These results demonstrate the effectiveness and robustness of the presented method, while the accuracy of the gaze estimation is still limited.

It should be noted that with the proposed method, we can perform gaze estimation on freely moving object as well as statically standing object. Figure 4.20 is an example of using our gaze estimation method on Data A (Section 4.4.1), a MAIKO performing traditional Japanese dance. The red arrow in Figure 4.20(b) and the red cross in Figure 4.20(c) illustrate the estimated gazing direction of the object. Since the full 3D shape



(a) Left eye, horizontal.

(b) Right eye, horizontal.

(c) Left eye, vertical.

(d) Right eye, vertical.

Figure 4.19: Gaze estimation errors of the proposed method. The upward and downward triangles at the bottom in each figure denote the signs (*i.e.* positive or negative) of the errors by the method with both the symmetry prior and the super-resolution image rendering technique. (*Copyright 2013 JSPS [80].*)

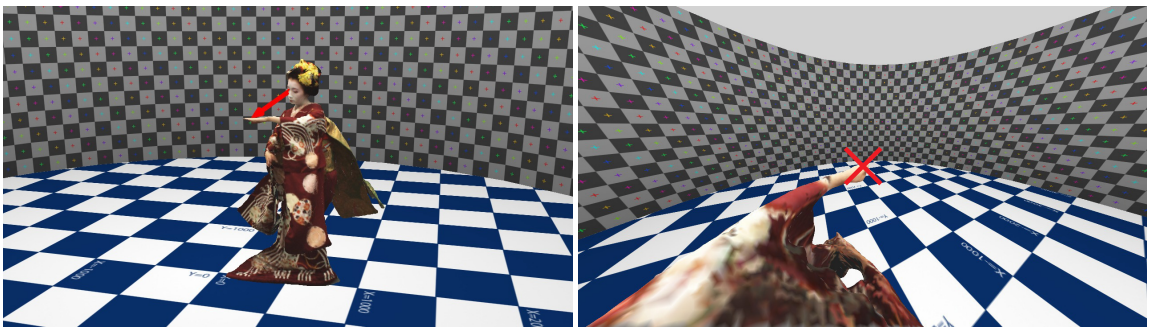
Table 4.3: Average gaze estimation errors of the proposed method.

	Left, horizontal	Right, horizontal	Left, vertical	Right, vertical
original	0.4326	0.4002	0.5321	0.5063
proposed	0.3297	0.3234	0.4610	0.4472

Table 4.4: Gaze estimation failures in 100 frames.

	Gaze estimation failure	Shape reconstruction failure	Iris detection failure
original	5	0	5
proposed	1	0	1

(a) Input multi-view images



(b) Estimated gazing direction in objective view (c) Estimated gazing direction in subjective view

Figure 4.20: Gaze estimation on freely moving object. (a) input multi-view images, (b) estimated gazing direction in objective view, (c) estimated gazing direction in subjective view. (*Copyright 2013 JSPS [80].*)

of the object is reconstructed (Figure 4.1 I), with the proposed method we can realize a subjective/first-person-view visualization, as is shown in Figure 4.20(c).

As for the gaze estimation results of Tobii X120 Eye Tracker, Figure 4.21 and 4.22 illustrate the angular errors of Tobii when working with markers inside and outside its effective region, respectively. The horizontal axis in Figure 4.21 denotes the frame IDs

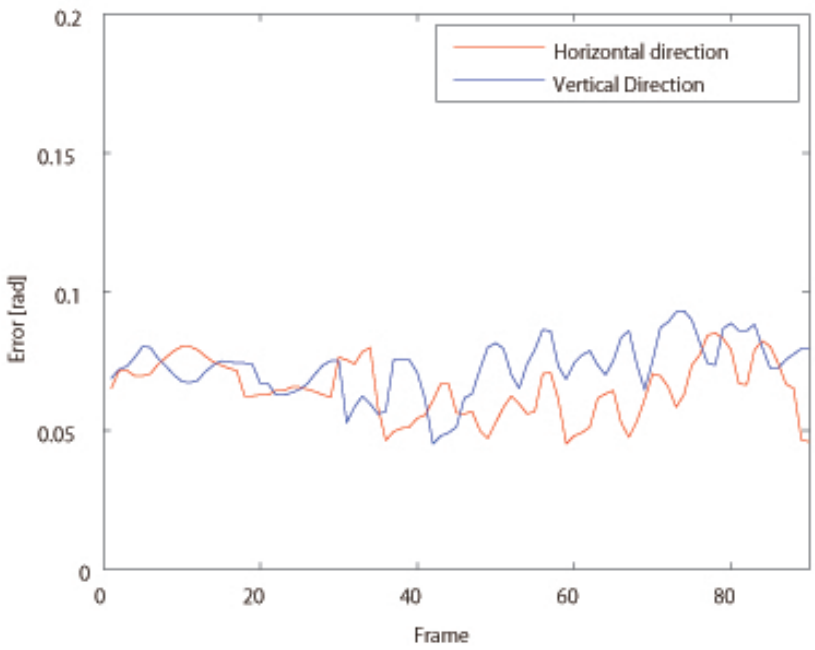


Figure 4.21: Gaze estimation errors of Tobii X120 Eye Tracker inside its effective region. (*Copyright 2013 JSPS [80].*)

where the subject is gazing at the five markers projected onto the wall, while the one in Figure 4.22 denotes the frame IDs where the subject is gazing at the two markers manually set outside Tobii's effective region, as is described in Section 4.4.2. Table 4.5 shows the average errors for Tobii X120 Eye Tracker. These results illustrate that Tobii performed the gaze estimation task with high accuracy when the markers are inside its effective region, while its accuracy dropped drastically when working outside its effective region. In addition, it should be noted that when the subject moved his head outside the freedom of head movement region, as is described in Table4.2, or rotated the head over about 40 degrees, the Tobii X120 Eye Tracker failed to track the subject's eyes and gave no results, while the proposed method still succeeded to perform the frontal image synthesis and gaze estimation process.

Taking into account all these experimental results we could conclude that the proposed method performs no better than conventional method under the situation that the object's head movement is strictly limited. However, our work showed its advantage in the capability of dealing with freely moving object, which makes it possible to estimate the gazing action of humans in natural and complicated activities.

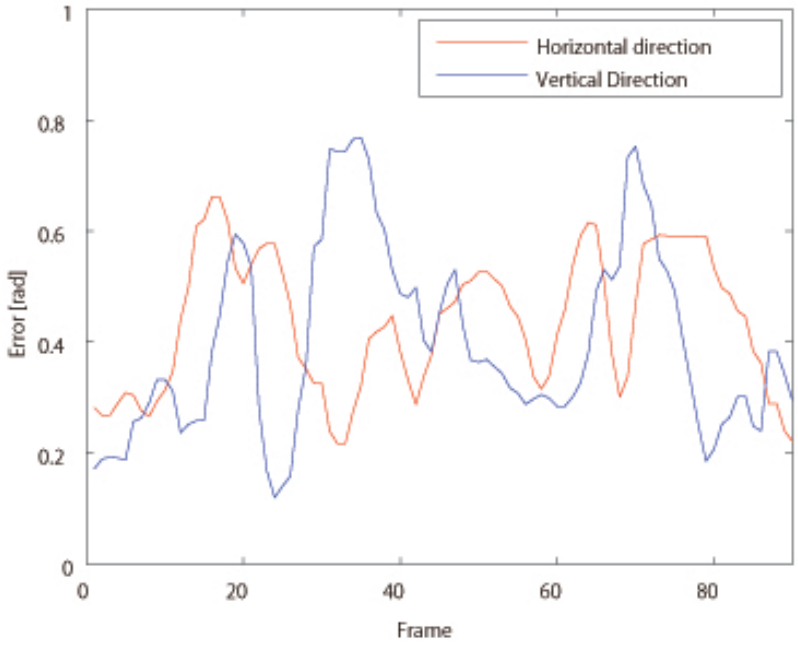


Figure 4.22: Gaze estimation errors of Tobii X120 Eye Tracker outside its effective region. (*Copyright 2013 JSPS [80].*)

Table 4.5: Average gaze estimation errors of Tobii X120 Eye Tracker.

	Horizontal	Vertical
Inside effective region	0.0651	0.0729
Outside effective region	0.4356	0.3996

4.5 Summary

In this chapter we proposed a novel 3D non-constrained and non-contact gaze estimation method that makes full use of the multi-view video data. The algorithm for the 3D gaze estimation consists of the 3D face area detection, the symmetry plane estimation, the accurate face surface reconstruction with the symmetry prior, the super-resolution frontal face image generation and the 3D gaze estimation based on the eyeball model.

Our algorithm worked stably to generate higher resolution frontal face images, and the accuracy of the last process to estimate the iris position and gaze direction was also improved. By comparing with a commercial gaze tracking device developed with conventional techniques, we have shown that our method exceeded others in the capability of performing robust gaze estimation on freely moving object.

With our gaze estimation method proposed in this chapter, the 3D video object's gazing direction (gaze vector) can be computed for each frame to be used to form an AHV, as introduced in Section 3.1.3 (Figure 3.3). Although the absolute accuracy of our gaze estimation method is still limited, it is adequate for modeling gaze actions in multi-party interaction event, which only requires a rough field of vision (FOV) of the object instead of a very precise gazing point.

Chapter 5

Multi-party Interaction Scene Editing Algorithm and Evaluation

In this section we present the spatiotemporal 3D editing algorithm using the AHV-based multi-party interaction modeling method. We first give a brief overview of our algorithm and the definition of related terms. Then we present each step of our action editing algorithm in details. Finally, an evaluation with real 3D video data is performed to prove the effectiveness of the proposed editing algorithm.

5.1 Multi-party Interaction Scene Editing Algorithm

5.1.1 Overview and definition of terms

With the proposed spatiotemporal editing algorithm, the input data is supposed to be a series of 3D mesh surfaces of the objects which can be frame-wise unstructured meshes where the temporal mesh coherency is not assumed. Each action sequence is considered as a collection of key segments and transitional segments, based on with or without contacts. For each short action scene formed up by a pair of key segments, there can exist multiple possible solutions of alignment, while an optimal solution for the entire interaction event could finally be found in the integration throughout the whole sequence.

As mentioned in Chapter 1, our multi-party interaction editing method consists of three steps: (1) data segmentation and temporal alignment, (2) intra-key segment editing by AHV-based constraint satisfaction and (3) inter-key segment global optimization. We first define editing-related terms as follows:

1. Action sequence: The whole sequence of each originally captured object, in the form of 3D surface meshes. Let $M = \{V, E\}$ denote a 3D mesh consisting of a

vertex set V and an edge set E, then an action sequence can be denoted as $S_i = \{M_s | s = 1, \ldots, n_s\}$.

2. Key segment: A subset of action sequence S_i, storing frames of interactive actions with contact (*e.g.* handshaking, hitting on the body, ...). We use $K_i = \{M_k | k = 1, \ldots, n_k \quad n_k < n_s\}$ to denote key segments. Since the objects are supposed to perform actions that match with each other, key segments should appear in pairs from different action sequences.

3. Transitional segment: Intermediate frames between key segments in the action sequence, in which interactive actions without contact are stored. We use $T_i = \{M_t | t = 1, \ldots, n_t \quad n_t < n_s\}$ to denote transitional segments, and it should be noted that there can be no transitional segment between two key segments.

4. Action Segment: A generic term for both key segment and transitional segment, denoted by $A_i = \{M_a | a = 1, \ldots, n_a \quad n_a < n_s\}$.

5. Interaction scene: Spatiotemporally synchronized short multi-party interaction scene synthesized from a single pair of key segments.

6. Interaction sequence: Spatiotemporally synchronized long multi-party interaction sequence synthesized by integrating all the key segment pairs and transitional segments in the original action sequences.

5.1.2 Action sequence segmentation and temporal alignment

Before performing action editing process, the original action sequences are required to be segmented into pairs of key segments and transitional segments. As described earlier, this action sequence segmentation is performed manually by the editor, through the following two steps.

1. Select the action frames containing interactive body actions with contact to be key segments, then those remaining action frames of interactive body actions without contact belong to the transitional segments. Since the original action sequences are supposed to match with each other, these key segments and transitional segments from multiple objects should appear in pairs.

2. For each pair of key segments, if body actions from each object inside this pair are unidirectional, keep it as is. Otherwise, further decompose it into multiple key seg-

ment pairs until all the body actions inside each key segment pair are unidirectional. This will ensure the simplicity of the AHV of body actions for future editing work.

Note that although our definition of AHV is capable of modeling reciprocating actions, that will complicate the modeling of synchronization between multiple AHVs. Besides, modeling reciprocating body actions inside single AHV will inevitably decrease the number of key segment pairs as well as the size of the solution group for each pair of key segments and thus, influence the artifact-smoothing-out capability of the global optimization process. With the segmentation process above we can ensure that the body actions inside each key segment pair are unidirectional so that the multi-party dictionary defined in Chapter 3 can be easily applied to perform the action editing process.

On the other hand, since the potential interactively corresponding actions in the original captured data would inevitably have temporal mismatches, action segments of each pair may also have different lengths of time duration. Therefore, we apply the following linear stretching to normalize the durations.

For a pair of action segment A_i^A and A_i^B,
if $\tau_i^A > \tau_i^B$,

$$M_{i_m}^{\prime A} = M_{i_n}^A \quad (0 < m < \tau_i^B, n = \left\lfloor m \times \frac{\tau_i^A}{\tau_i^B} \right\rfloor), \tag{5.1}$$

otherwise,

$$M_{i_m}^{\prime B} = M_{i_n}^B \quad (0 < m < \tau_i^A, n = \left\lfloor m \times \frac{\tau_i^A}{\tau_i^B} \right\rfloor), \tag{5.2}$$

where τ_i^A and τ_i^B represent the duration of A_i^A and A_i^B, $M_{i_n}^A$ represents the nth frame in A_i^A, and $M_{i_m}^{\prime A}$ represents the mth frame of the normalized A_i^A. $\lfloor x \rfloor$ is the floor function that returns the largest integer not greater than x. Note that the same processing will be applied to both the key segment pairs and the transitional segment pairs.

5.1.3 Intra-key segment editing

Editor defined constraints

By performing the data segmentation and temporal alignment, multiple objects' interactive body actions with contact are organized into key segment pairs, and inside each pair objects' body actions are scaled into the same temporal length. Then the AHVs of each object's body action in each segment pair can be computed as $v_{\tau_i}^A(x, y, z, t)$ and $v_{\tau_i}^B(x, y, z, t)$.

For each interaction event inside a key segment pair, we model that the two interacting objects should follow certain spatiotemporal constraints based on (1) body action interacting type, (2) mutual visibility and (3) fidelity requirement.

(1) Body action interacting type constraint:

First for each key segment pair the interactive contact type of objects' body actions should be decided by the editor based on his/her editing intension, producing one type of spatiotemporal constraint from the interaction dictionary.

(2) Mutual visibility constraint:

Second, for multi-party interaction scenes, usually the objects should be visible for each other (Fig. 5.1). We first compute the object's gazing direction $\overrightarrow{G_i}$ for each frame using the method introduced in Chapter 4. Then as discussed in Section 3.1.3, a gazing cone can be generated for each object at each frame. With these gazing cones we can create the AHV of each object's interactive zone, denoted by $g^A_{\tau_i}(x,y,z,t)$ and $g^B_{\tau_i}(x,y,z,t)$. Then the mutual visibility constraint can be defined as: one object's AHV of the gazing cone $g^A_{\tau_i}(x,y,z,t)$ and the other object's AHV of the body action $v^B_{\tau_i}(x,y,z,t)$ fulfill the ⓪ & ⓪-(a) type from the multi-party interaction dictionary. The detailed mathematical constraint is described as follows:

$$\bigcap_{t_i=0}^{\tau_i-1}((G'^A_{t_i}(x,y,z)\cap V'^B_{t_i}(x,y,z))\cap(G'^B_{t_i}(x,y,z)\cap V'^A_{t_i}(x,y,z)))\neq\phi. \quad (5.3)$$

Here $G'^A_{t_i}(x,y,z)$ represents a subset of $g^A_{\tau_i}(x,y,z,t)$'s spatial volume $G^A_i(x,y,z)$ at time t, and $V'^B_{t_i}(x,y,z)$ represents a subset of $v^B_{\tau_i}(x,y,z,t)$'s spatial volume $V^B_i(x,y,z)$ at time t

It should be noted that the visibility constraint is an optional constraint for the editor, meaning that it is not always required for each moment in the key segment pair.

(3) Fidelity constraint:

Third, the fidelity requirement avoids unnatural fakeness, such as two objects merge into each other's body. Let $v^{\bar{A}}_{\tau_i}(x,y,z,t)$ and $v^{\bar{B}}_{\tau_i}(x,y,z,t)$ denote the AHVs of the objects that should not contact in the interaction event, then the fidelity constraint can be defined as: $v^{\bar{A}}_{\tau_i}(x,y,z,t)$ and $v^{\bar{B}}_{\tau_i}(x,y,z,t)$ fulfill the ⓪ & ⓪-(c) type from the multi-party interaction dictionary. The detailed mathematical constraint is de-

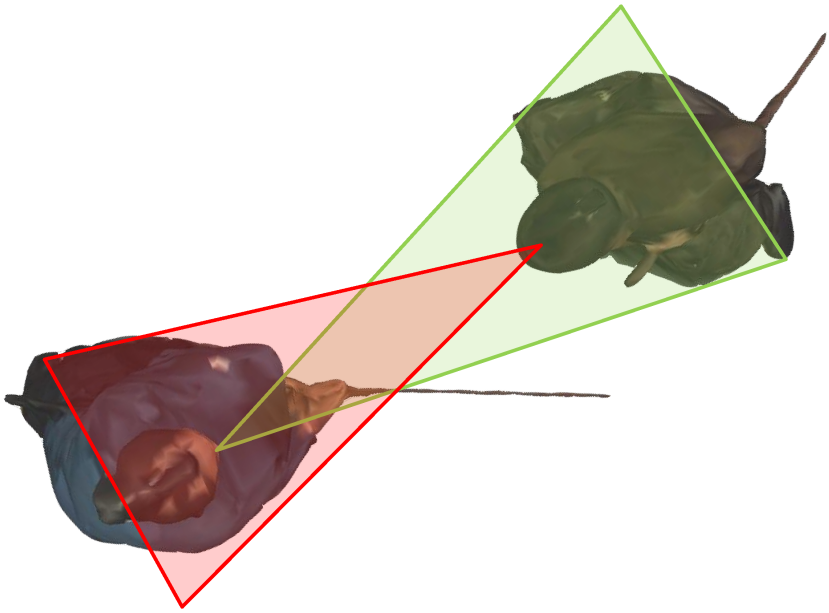


Figure 5.1: Mutual visibility constraint.

scribed as follows:

$$\begin{aligned} &\forall\, v^{\bar{A}}_{\tau_i} \underset{(x,y,z)}{\cap} v^{\bar{B}}_{\tau_{i'}}, \\ &\bigcup_{i=1,i'=1}^{i=n,i'=n'} ([t^{\bar{A}}_{\mathrm{start}_i}, t^{\bar{A}}_{\mathrm{end}_i}] \cap [t^{\bar{B}}_{\mathrm{start}_{i'}}, t^{\bar{B}}_{\mathrm{end}_{i'}}]) = \phi. \end{aligned} \tag{5.4}$$

Action scene editing using constraint satisfaction

With the editor defined constraints, we compute acceptable relative positions of multiple objects by constraint satisfaction.

For each action scene consisting of a pair of key segments K_i^A and K_i^B, first multiple objects are put into the same world coordinate system. Let C_i^A, C_i^B denote the 2D positions of the objects's AHV root nodes on the ground plane, $L_i^{AB} = |\overrightarrow{C_i^A C_i^B}|$ denote the distance between multiple objects, and let θ_i^A, θ_i^B represent the included angles between

the average viewing direction of each AHV and line $C_i^A C_i^B$, then the relative positions between multiple objects can be represented by $P_i^{AB} = (L_{i_n}^{AB}, \theta_{i_n}^A, \theta_{i_n}^B)$.

The editing work for a single action scene is to find a sub-space $R_i^{AB} = (L_{i_j}^{AB}, \theta_{i_j}^A, \theta_{i_j}^B) \subset P_i^{AB}$, in which parameters ensure that the objects completely fulfill the editor defined constraints.

We solve this problem by a generation-and-test approach. We sample parameters in P_i^{AB} space, and test if the synthesized segment satisfies the editor-defined constraints or not. Hence, the acceptable parameter space R_i^{AB} is descretized as a set of sampled points, each of which represents an acceptable editing result for this segment pair.

In order to increase the computational efficiency of the constraint satisfaction process, we introduce a coarse-to-fine strategy that tries to reduce the amount of samples to be tested by using a multi-level processing.

Step 1 Firstly the P_i^{AB} space is decomposed into N levels, and the sampling resolution of level n $(1 < n < N)$ is 2^{n-1} times lower than the finest resolution $(Res(L), Res(\theta), Res(\theta))$.

Step 2 Next, the editor-defined constraints are tested for all the samples in $level\ n = N$, discarding the parameters that do not fulfill the constraints and getting a solution group $R_{i^N}^{AB}$.

Step 3 In turn, the same test will be repeated for $level\ (n-1) = (1 < n <= N)$ sampling space, and only the samples whose distances to their nearest points in $R_{i^n}^{AB}$ are no larger than the sampling resolution of the current level will be tested.

By repeating such process down to $level\ n = 1$, the samples of possible solutions required to be tested would be drastically reduced, as illustrated in Section 5.2.

The cost of this step is $O(N_k \cdot s)$, where N_k is the number of key-segment pairs, and s is the number of samples in the parameter space.

5.1.4 Inter-key segment optimization

The intra-key segment editing process computes the solution groups for each key segment pair. Then we will continue to combine them together with the transitional segments in order to synthesize a complete multi-party interaction sequence, by (1) finding the optimal solution in each key segment pair and (2) adding in the transitional segments and performing path optimization.

Optimal solution searching for key segments

In the original data before editing, we denote the 2D positions of the AHVs' root node on the ground plane in the world coordinate system with $C_1^A, C_2^A, \ldots, C_I^A, C_1^B, C_2^B, \ldots, C_I^B$, and we consider the object's average gazing directions from each frame as the facing directions of the AHVs in the world coordinate system, denoted by $\overrightarrow{F_1^A}, \overrightarrow{F_2^A}, \ldots, \overrightarrow{F_I^A}, \overrightarrow{F_1^B}, \overrightarrow{F_2^B}, \ldots, \overrightarrow{F_I^B}$. In order to make natural and smooth editing preserving the original action, we should maintain the vectors $\overrightarrow{C_i^A C_{i+1}^A}$ and $\overrightarrow{C_i^B C_{i+1}^B}$ (i=1,...,I-1) as much as possible to minimize the artificial offsets. As well, the facing directions $\overrightarrow{F_i^A}$, $\overrightarrow{F_i^B}$ should also be preserved. If we denote the edited ideal positions of the AHVs' root node as $\bar{C}_1^A, \bar{C}_2^A, \ldots, \bar{C}_I^A, \bar{C}_1^B, \bar{C}_2^B, \ldots, \bar{C}_I^B$, the edited facing directions of the AHVs as $\overrightarrow{\bar{F}_1^A}, \overrightarrow{\bar{F}_2^A}, \ldots, \overrightarrow{\bar{F}_I^A}, \overrightarrow{\bar{F}_1^B}, \overrightarrow{\bar{F}_2^B}, \ldots, \overrightarrow{\bar{F}_I^B}$, and the angles between $\overrightarrow{\bar{F}_i^A}$ and $\overrightarrow{\bar{C}_i^A \bar{C}_i^B}$ as $\bar{\theta}_i^A$, then we should search for the optimal solution for each key segment pair by fulfilling:

$$\begin{aligned} \min \sum_{i=1}^{I-1} [& |\overrightarrow{C_i^A C_{i+1}^A} - \overrightarrow{\bar{C}_i^A \bar{C}_{i+1}^A}|^2 + |\overrightarrow{C_i^B C_{i+1}^B} - \overrightarrow{\bar{C}_i^B \bar{C}_{i+1}^B}|^2 \\ & + \lambda(|\overrightarrow{F_i^A} - \overrightarrow{\bar{F}_i^A}|^2 + |\overrightarrow{F}_i^B - \overrightarrow{\bar{F}_i^B}|^2)], \end{aligned} \tag{5.5}$$

and

$$|\overrightarrow{\bar{C}_i^A \bar{C}_i^B}| = L_i^{\bar{A}\bar{B}} = L_{i_j}^A B = |\overrightarrow{C_{i_j}^A C_{i_j}'^B}|, \tag{5.6}$$

$$\bar{\theta}_i^A = \theta_i^A, \bar{\theta}_i^B = \theta_i^B, \tag{5.7}$$

where λ is a weighting factor to balance the translational and rotational artifacts. The optimal solution for all the key segments can be computed by solving function (5.5), (5.6) and (5.7) using dynamic programming approach.

The cost of this process is $O((N_{k-1}-1){n_i}^2)$, where N_k is the number of key-segment pairs, and n_i is the average solution number for each key-segment pair.

Path optimization for transitional segments

Having found the optimal solutions for all the key segments, we then need to add in the transitional segments and perform a path optimization to smooth out the offsets caused by the intra-key segment editing process.

Suppose inside a transitional segment, the original positions of the object's center of masses for all the frames are $C_1, C_2, \ldots, C_{m-1}$ and the edited positions are denoted as $\bar{C}_1, \bar{C}_2, \ldots, \bar{C}_{m-1}$. Original and edited facing directions are denoted respectively as

$\overrightarrow{F_1}, \overrightarrow{F_2}, \ldots, \overrightarrow{F_{m-1}}$ and $\overrightarrow{\bar{F}_1}, \overrightarrow{\bar{F}_2}, \ldots, \overrightarrow{\bar{F}_{m-1}}$. Then we define our objective function for performing path optimization as follows:

$$\begin{aligned} f = \sum_{i=1}^{m-1} [\delta f_a(\bar{C}_i) + (1-\delta)\omega_i^C f_v(\bar{C}_i) \\ +\lambda(\delta f_a(\overrightarrow{\bar{F}_i}) + (1-\delta)\omega_i^F f_v(\overrightarrow{\bar{F}_i})]. \end{aligned} \tag{5.8}$$

Here δ and λ are weighting factors to balance the acceleration and speed, the translation and rotation, respectively. Specifically, $f_a(\bar{C}_i)$ and $f_a(\overrightarrow{\bar{F}_i})$ is defined as follows to preserve the original accelerations for each frame:

$$f_a(\bar{C}_i) = |(C_{i-1} - 2C_i + C_{i+1}) - (\bar{C}_{i-1} - 2\bar{C}_i + \bar{C}_{i+1})|^2, \tag{5.9}$$

$$f_a(\overrightarrow{\bar{F}_i}) = |(\overrightarrow{F_{i-1}} - 2\overrightarrow{F_i} + \overrightarrow{F_{i+1}}) - (\overrightarrow{\bar{F}_{i-1}} - 2\overrightarrow{\bar{F}_i} + \overrightarrow{\bar{F}_{i+1}})|^2, \tag{5.10}$$

and $f_v(\bar{C}_i)$, $f_v(\overrightarrow{\bar{F}_i})$ is defined as follows to preserve the original speed for each frame:

$$f_v(\bar{C}_i) = |\frac{C_{i-1} - C_{i+1}}{2} - \frac{\bar{C}_{i-1} - \bar{C}_{i+1}}{2}|^2, \tag{5.11}$$

$$f_v(\overrightarrow{\bar{F}_i}) = |\frac{\overrightarrow{F_{i-1}} - \overrightarrow{F_{i+1}}}{2} - \frac{\overrightarrow{\bar{F}_{i-1}} - \overrightarrow{\bar{F}_{i+1}}}{2}|^2. \tag{5.12}$$

Besides,

$$\omega_i^C = \frac{1}{1 + \exp\{\alpha(v_i - v_k)\}}, \tag{5.13}$$

$$\omega_i^F = \frac{1}{1 + \exp\{\alpha(r_i - r_k)\}}, \tag{5.14}$$

$$v_i = \frac{|C_{i-1} - C_i| + |C_i - C_{i+1}|}{2}, \tag{5.15}$$

$$r_i = \frac{|\overrightarrow{F_{i-1}} - \overrightarrow{F_i}| + |\overrightarrow{F_i} - \overrightarrow{F_{i+1}}|}{2}, \tag{5.16}$$

where v_k and r_k are constant parameters to be set by editors.

Minimizing this objective function is a non-linear optimization of $2m$ parameters for positions and m parameters for rotation. By applying this process for each object in each transitional segment, we can perform path optimization onto all the transitional segments

Table 5.1: Specifications of the 3D video studios used in this research.

	3D Video Studio A	3D Video Studio B
Shape	Rounded square	Dodecagon
Size	6 m diameter, 2.5 m	6 m diameter, 2.4 m height
Camera	single ring with ceiling cameras arrangement	high and low double rings with ceiling cameras
Camera	Pointgrey GRAS-20S4C $\times$ 16	Sony XCD-X710CR $\times$ 15
Imager	1/1.8 inch 1CCD	1/3 inch 1CCD
Image format	UXGA/RAW	XGA/RAW
Lens	C-mount, 6 mm & 3.5 mm	C-mount, 6 mm & 3.5 mm
Frame rate	25 fps	25 fps
Capture PC	2	15
Connection	IEEE 1394b 10 m cable	IEEE 1394a 20 m cable
Datarate	45.78 MB/s(366MB/s per PC)	18.75 MB/s
Background	Green screen	Gray plywood
Lighting	Overhead inverter fluorescent lights	Overhead inverter fluorescent lights

and finally, acquire a natural looking spatiotemporally synchronized multi-party interaction 3D video sequence for free-viewpoint browsing.

5.2 Experiment and Evaluation

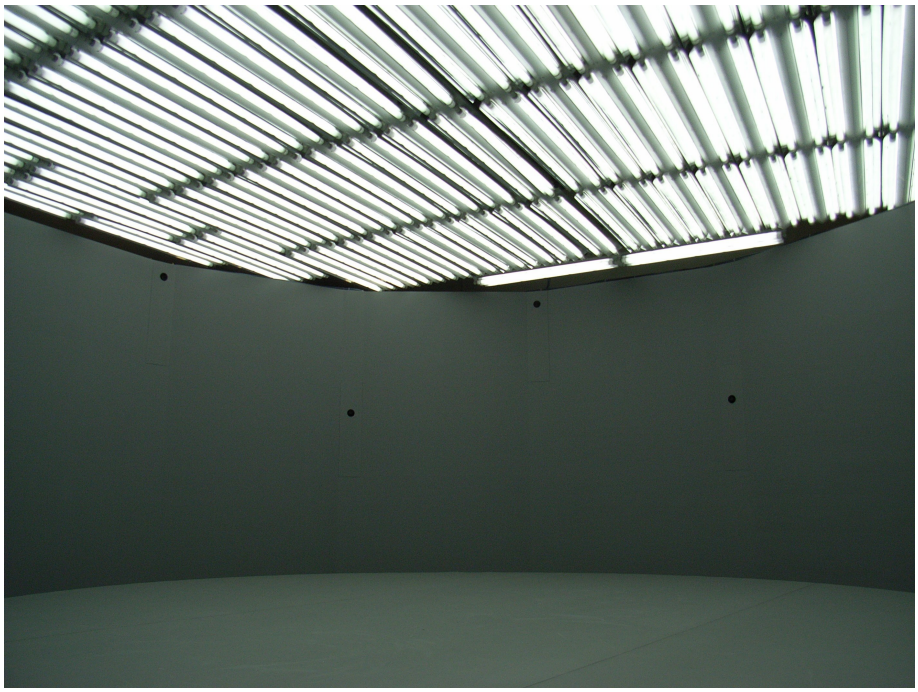
In this section we present an evaluation of the proposed multi-party interaction editing method with real 3D video data.

5.2.1 Experiment setup and pre-processing

In order to prove the effectiveness of the proposed method, we prepared four sets of real 3D video data (Handshaking, Highfive, Chambara, Kungfu) captured from two studios with different setups. Each of them includes a multi-party interaction sequence performed by two actors, separately captured by 16/15 calibrated UXGA/XGA cameras running at 25 Hz with 1 msec shutter speed. Fig. 5.2 shows an overview of the 3D video studios being used for this evaluation. The detailed parameters of the studios are listed in Table 5.1. Fig. 5.3 shows the original separately captured image data of multiple objects. Note that the Chambara and Kunfu data are captured using studio A (Fig. 5.2(a)), while the Handshaking and Highfive data are captured under another studio setup B (Fig. 5.2(b)).

In the Handshaking and Highfive datasets, objects perform simple daily communicative actions of shaking hands and clapping hands. While in the Chambara dataset, the two separately captured actors are performing a complex pre-designed sword fighting action

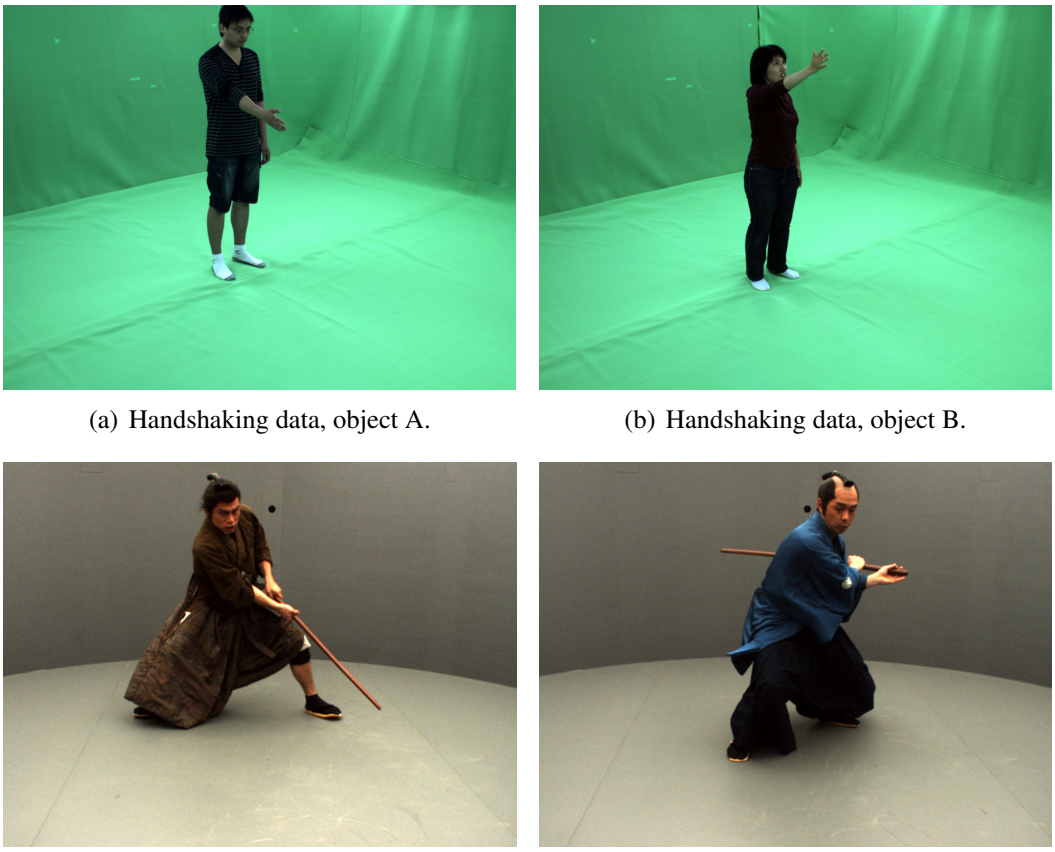
(a) Studio A.



(b) Studio B.

Figure 5.2: Two different 3D video studios used in experiments. (*Copyright 2014 IEEE.*)

sequence, and in the Kungfu dataset, a pre-designed Kungfu fighting sequence is separately performed by two actors as the multi-party interaction events. These four sets of



(a) Handshaking data, object A.

(b) Handshaking data, object B.

(c) Chambara data, object A.

(d) Chambara data, object B.

Figure 5.3: Separately captured image data of multiple objects from two different studios. (*Copyright 2014 IEEE.*)

3D video data are reconstructed with frame-wise 3D shape reconstruction method proposed by Nobuhara *et al.* [81]. And Takai *et al.*'s appearance based view-independent texture generation technique [46] is introduced for recovering the surface textures of the data. It should be noted that the 3D video data does not contain a unified 3D mesh model. Skeleton models inside objects' surfaces are not available either.

As for pre-processing, each dataset is manually segmented into key segment and transitional segment pairs, following the criteria described in Secion 5.1.2. Inside each action segment pair, actions of multiple objects are aligned into the same temporal length. Table 5.2 illustrates the frame number of each object before and after temporal alignment. The number of key segments and frame numbers in each key segments are listed in the left five columns of Table 5.3.

Table 5.2: Temporal alignment result.

	Handshaking-A	Handshaking-B	Highfive-A	Highfive-B
before	24	29	10	15
after	23	23	9	9
	Chambara-A	Chambara-B	Kungfu-A	Kungfu-B
before	602	500	749	575
after	425	425	550	550

5.2.2 Qualitative evaluation

To demonstrate the effectiveness of the proposed method, we first perform a qualitative evaluation by visually observing the editing results. Figs. 5.4(a)-(h) show the editing results of the Handshaking and Highfive dataset. In the editing results multiple objects actions properly match with each other.

For the two more complex dataset, Chambara and Kunfu, we conduct the evaluation by comparing the editing results by our method (Figs. 5.5(e)-(h), Figs. 5.6(e)-(h)) with the results by a baseline method (Figs. 5.5(a)-(d), Figs. 5.6(a)-(d)) in which only the time-duration is aligned and an initial relative position of multiple objects is given in original data. Note that the baseline method only puts the original data into the same coordinate system and manually gives an initial relative position of the separately captured data. It does not contain any more spatial editing, so the action dynamics of its results is nearly the same as the original data.

As illustrated in Figs. 5.5(a)-(d) and Figs. 5.6(a)-(d), lots of spatial mismatches exist in the synthesized 3D scenes. On the other hand, the editing results using the proposed method are illustrated in Figs. 5.5(e)-(h) and Figs. 5.6(e)-(h). It can be clarified that in each synthesized interaction scene, the relative location of the two objects look natural and reasonable, as well they well qualify the editor defined spatiotemporal constraints.

Besides, Fig. 5.7 shows multi-view rendering results of a "*hit on target*" scene from the Chambara dataset, with and without performing the proposed method. In Figs. 5.7(a), 5.7(b), 5.7(e) and 5.7(f), the rendered images look acceptable for both the results by the basline method and the results by the proposed method. However, in Figs. 5.7(c), 5.7(d), 5.7(g) and 5.7(h), the results by the baseline method look very fake since the attacker failed in really cutting into the opponent's body, while the results by our method still look natural and sound. It can be seen from these figures that although the synthesized scene without spatial editing may look natural from certain specific views, the arbitrary view

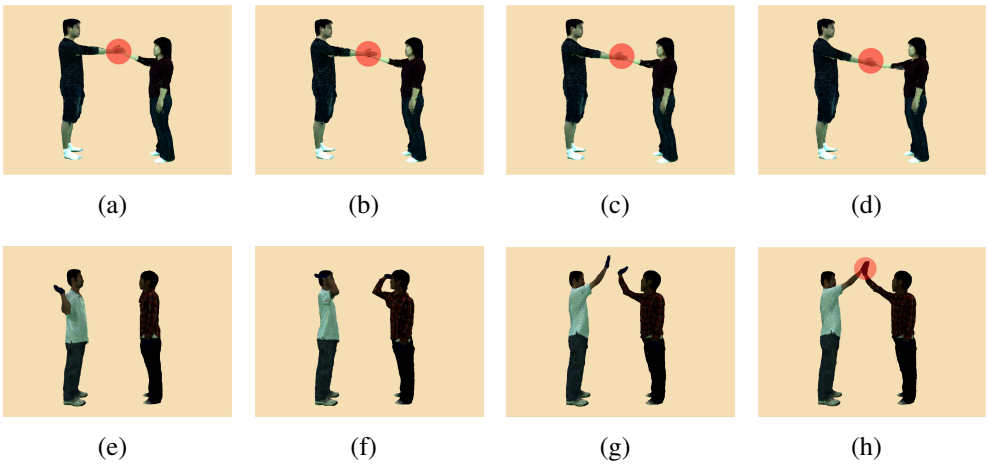


Figure 5.4: Spatiotemporal editing results for Handshaking and Highfive data. The red circles are superposed to highlight the editing results. (*Copyright 2014 IEEE.*)

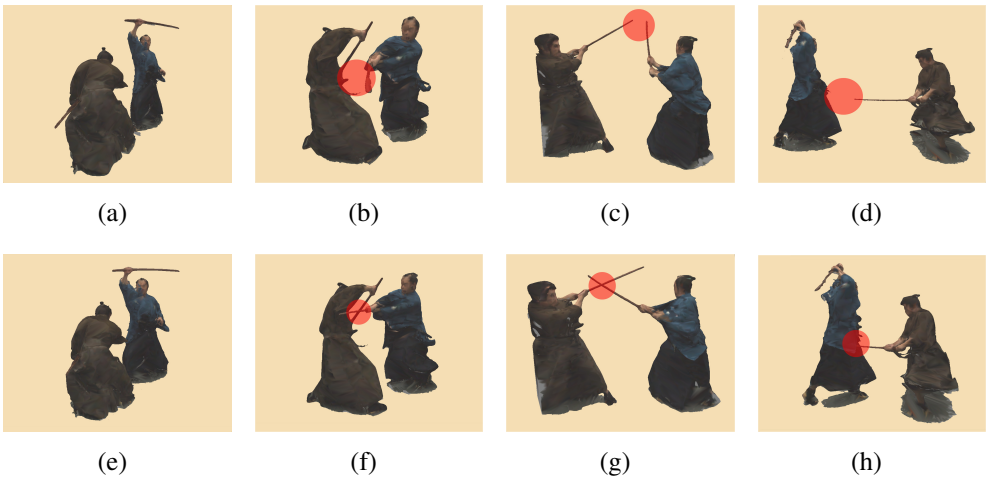
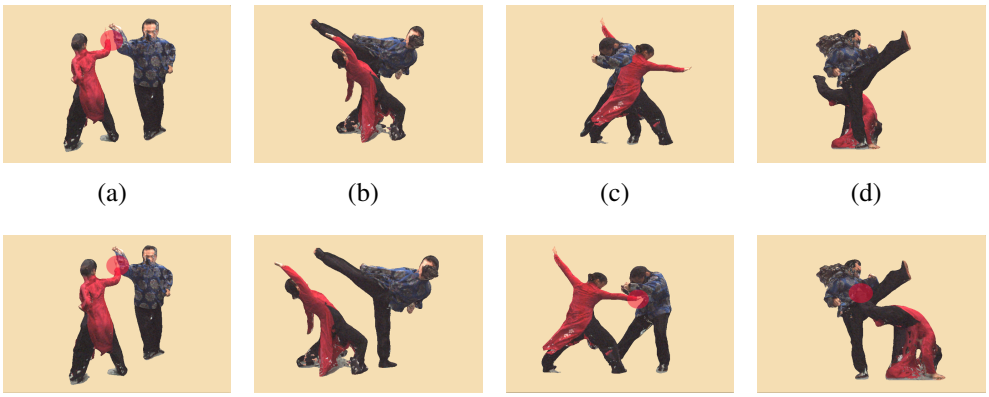


Figure 5.5: Spatiotemporal editing results for Chambara data. Top: Editing results of the baseline method. Bottom: Editing results of the proposed method. The red circles are superposed to highlight the editing results. (*Copyright 2014 IEEE.*)

point browsing nature of 3D video requires the view-independent fidelity consistency that only the editing results by the proposed method fulfill.

In addition, Fig. 5.8 illustrates the first-person-view images rendered from the editing results, using Shi *et al.*'s method [75]. It can be confirmed that in each editing result the two objects are inside each other's view, fulfilling the mutual visibility constraint.

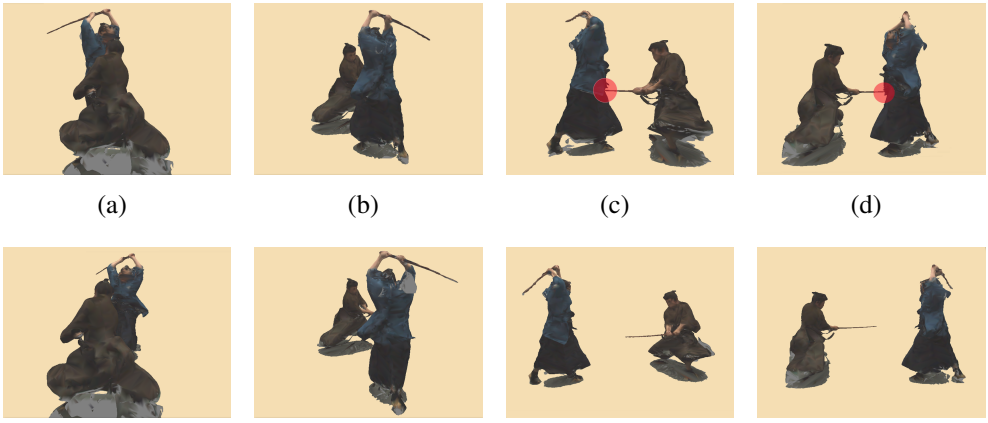
On the other hand, Fig. 5.9 illustrates the editing results of the attack and guard scene, partially using the user defined constraints. Fig. 5.9(a) is the editing result only using the mutual visibility constraint, in which the two objects are soundly inside each other's



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5.6: Spatiotemporal editing results for Kungfu data. Top: Editing results of the baseline method. Bottom: Editing results of the proposed method. The red circles are superposed to highlight the editing results. (*Copyright 2014 IEEE.*)



(a) (b) (c) (d)

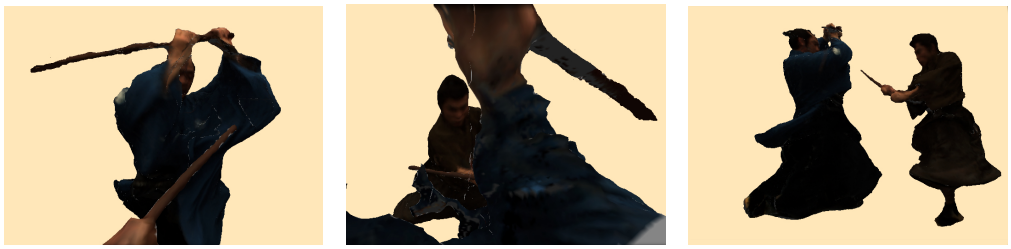
(e) (f) (g) (h)

Figure 5.7: Multi-view rendering results of a “hit on target" scene. Top: Editing results of the baseline method. Bottom: Editing results of the proposed method. The red circles are superposed to highlight the editing results. (*Copyright 2014 IEEE.*)

eye sight, but their weapons are not contacting with each other as they should be. While Fig. 5.9(b) is the editing result only using the mutual visibility constraint, in which the weapons are correctly contacting, but object B is outside object A's field of view. These results show that each of the user defined constraints can partially ensure the naturalness of the editing results, but in many multi-party interaction scenes they need to be used together to generate the most natural looking editing results.

(a) Handshaking data, object A. (b) Handshaking data, object B. (c) Handshaking data, corresponding third-person-view.

(d) Highfive data, object A. (e) Highfive data, object B. (f) Highfive data, corresponding third-person-view.



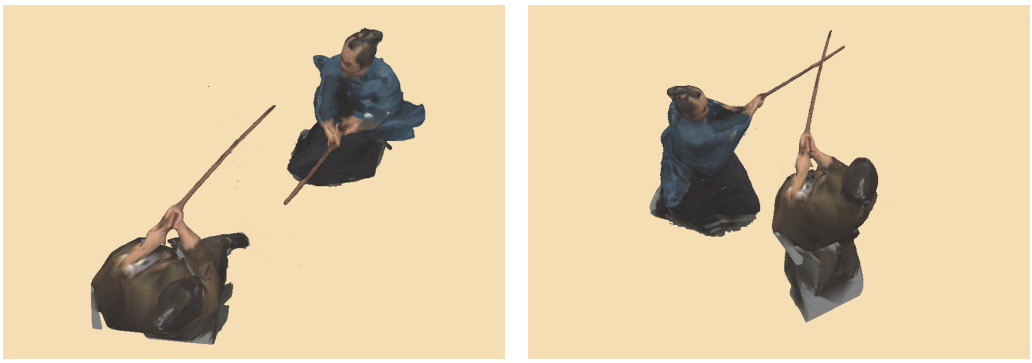
(g) Chambara data, object A. (h) Chambara data, object B. (i) Chambara data, corresponding third-person-view.

(j) Kungfu data, object A. (k) Kungfu data, object B. (l) Kungfu data, corresponding third-person-view.

Figure 5.8: First-person-view images rendered from the editing result. (*Copyright 2014 IEEE.*)

5.2.3 Quantitative evaluation

Quantitative evaluations are performed for the Handshaking, Chambara and Kunfu data, respectively. For the computation of intra-key segment constraint satisfaction, parameters are set as: $Res(L) = 10\ mm$, which equals the resolution of the AHV, and $Res(\theta) =$



(a) Editing result only using mutual visibility constraint.

(b) Editing result only using interactive contact type constraint.

Figure 5.9: Editing results using single spatiotemporal constraint. (*Copyright 2014 IEEE.*)

2 *degree*, $N = 3$. Note that when $N > 3$, faster computational speed can still be achieved, but in the final results the optimal solution is lost, due to the reason that the initial sampling resolution is too low. Computations without coarse-to-fine strategy are also tested as a comparison, and the corresponding results are listed inside brackets in Table 5.3 and Table 5.5.

In the editing results of the Handshaking data, for each frame the hands of multiple objects contact with each other. However, in a real handshaking situation, the two objects' palms should stick to each other without any relative movements. Without skeleton model based posture editing, we can only minimize the total relative movements between the two objects' palms. For each frame in the editing results, the center of mass of each object's hand is computed. Then the variance of the distances between the two centers of masses from each frame in the editing results reflects the degree of relative movements between objects palms. In the editing result of the proposed method, the computed average distance is 14.35 mm, variance is 28.32 mm^2, which implies that the unnatural artifacts of relative movements between palms in the editing result are rather small.

For Chambara and Kungfu data, in both Figs. 5.10 and 5.11, four trajectories on the floor plane are compared: object A before editing, object A after editing, object B before editing and object B after editing. They are drawn in different colors, and the dotted lines and full lines represent the trajectories before and after editing, respectively. Besides, the small colored squares on each curve denote every 50 frames of the data. We also give an example of the objects' positions and the corresponding rendered images of the synthesized results.

It can be seen in these figures that the global locations of the trajectories have changed

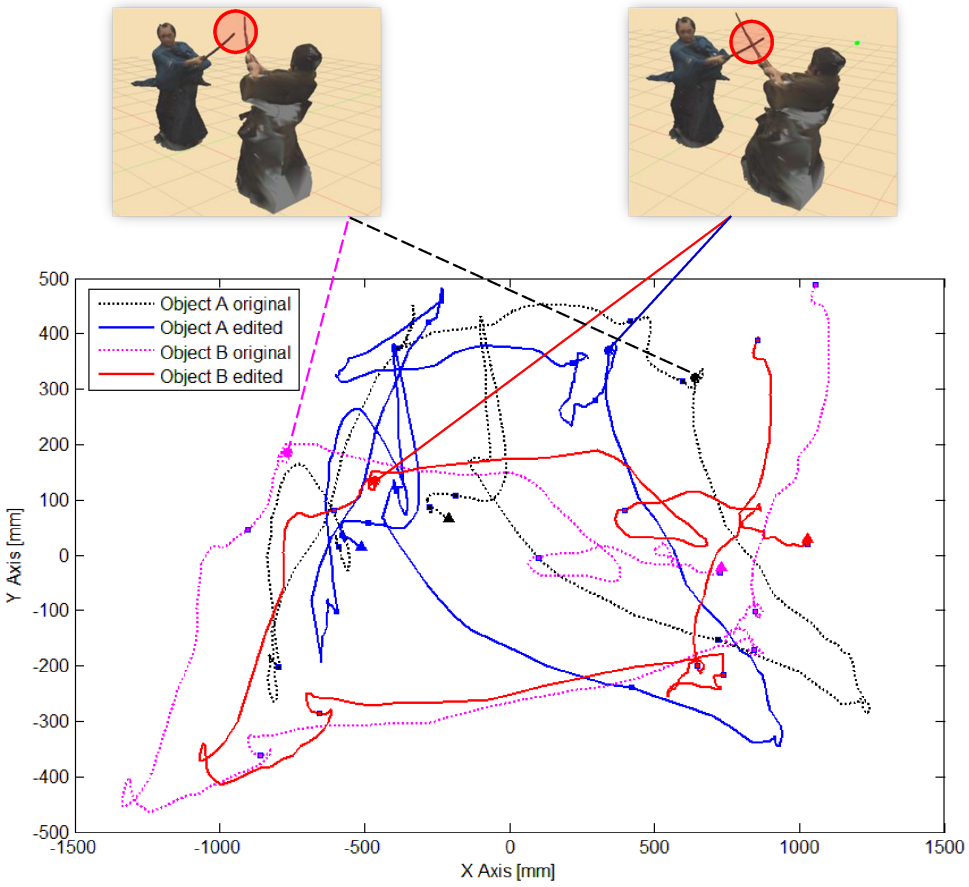


Figure 5.10: Spatiotemporal editing result for Chambara dataset. (*Copyright 2014 IEEE.*)

distinctively, while the local shapes of the trajectories before and after editing still look similar to each other, meaning that the action dynamics in the original data have been kept well in the editing process.

Table 5.4 shows a comparison between the baseline data and the editing result, considering seven components: the average difference of position, the mean-squared-difference (MSD) of position, the average translation velocity (baseline/proposed method), the average difference of translation velocity, the mean-squared-difference of translation velocity and angular velocity, the average rotation speed (baseline/proposed method), the average difference of rotation speed and the mean-squared-difference of rotation speed. Here the mean-squared-difference is computed by:

$$\mathrm{MSD} = \frac{1}{N}\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2, \tag{5.17}$$

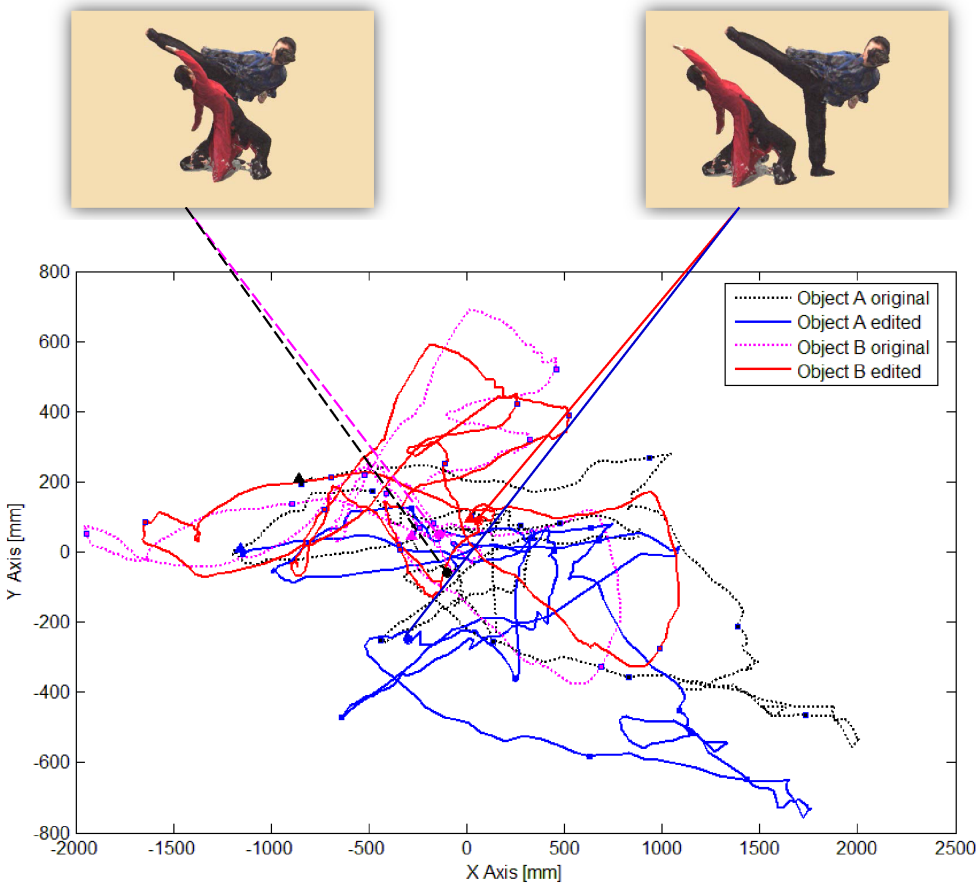


Figure 5.11: Spatiotemporal editing result for Kungfu dataset. (*Copyright 2014 IEEE.*)

where N is the total frame number of the data, $\hat{Y}$ is a vector of the editing results by the proposed method, and Y is a vector of the results by the baseline method. The units for positions, translation velocities and rotation speeds are mm, mm/frame and degree/frame.

Table 5.4 shows that after editing, the spatial positions of the data have been changed drastically, while the changes on translation velocity and rotation speed are relatively small, meaning that the action dynamics of the original data have been preserved well.

The right two columns of Table 5.3 illustrates the intra-key segment editing solution numbers for each key segment pair. Thanks to the advantage of the AHV based interaction dictionary, we can easily apply constraint satisfaction to automatically compute all the reasonable solutions for each short action scene. Also the completeness of these solution groups makes it possible to find an optional solution with minimum artifacts through computation for the whole interaction sequence. As is shown in the table, each key segment pair has hundreds or even thousands of solutions, and approximately 10^{25} to 10^{42}

Table 5.3: Intra-key segment editing results.

#Key	Chambara FPK	Kungfu FPK	Handshaking FPK	Highfive FPK
01	18	11	12	9
02	9	13	11	-
03	7	8	-	-
04	8	15	-	-
05	12	9	-	-
06	8	7	-	-
07	16	12	-	-
08	11	13	-	-
09	17	10	-	-
10	-	9	-	-
11	-	13	-	-
12	-	6	-	-
13	-	10	-	-
14	-	11	-	-
15	-	14	-	-
#Key	Chambara SPK	Kungfu SPK	Handshaking SPK	Highfive SPK
01	799 (1883)	294 (567)	56 (89)	184 (317)
02	337 (448)	303 (493)	59 (103)	-
03	203 (372)	427 (1038)	-	-
04	471 (616)	513 (821)	-	-
05	498 (1017)	108 (357)	-	-
06	63 (228)	254 (460)	-	-
07	157 (391)	377 (713)	-	-
08	292 (563)	296 (553)	-	-
09	506 (1020)	538 (937)	-	-
10	293 (634)	-	-	-
11	275 (706)	-	-	-
12	-	189 (397)	-	-
13	-	316 (690)	-	-
14	-	1031 (1697)	-	-
15	-	311 (542)	-	-

sets of final results can be generated for each dataset based on the combination of these solutions. Without the proposed method it is almost intractable to manually find the best combination from each solution group and achieve the optimal solution.

Table 5.4: Quantitative evaluation of the editing results.

Data Sequence	Avg. DOP	MSD of DOP	Avg. TV (b/p)	Avg. DOTV
Chambara-A	257.01	69457	15.86 / 17.15	1.29
Chambara-B	263.10	73582	17.09 / 16.22	-0.96
Kungfu-A	241.75	62408	28.53 / 30.46	1.93
Kungfu-B	237.50	58317	21.10 / 22.98	1.88
Data Sequence	MSD of DOTV	Avg. RV (b/p)	Avg. DORV	MSD of DORV
Chambara-A	15.76	2.63	0.20	0.09
Chambara-B	8.58	3.22	0.24	0.12
Kungfu-A	7.47	4.09	0.18	0.12
Kungfu-B	6.48	5.53	0.21	0.16

Table 5.5: Processing cost of the proposed method.

Data Sequence	Avg. MHV computation	Avg. Constraint satisfaction
Chambara	59 sec	67 (203) sec
Kungfu	61 sec	63 (163) sec
Handshaking	23 sec	37 (158) sec
Highfive	21 sec	54 (166) sec
Data Sequence	Optimal solution searching	Path optimization
Chambara	21 (52) min	382 sec
Kungfu	35 (64) min	401 sec
Handshaking	34 (61) sec	- sec
Highfive	- min	-sec

5.2.4 Processing cost of the proposed method

The experiment is conducted using a PC with Intel(R) Core(TM)2 Duo CPU @ 3.00GHz. Under the finest sampling resolution of 10 mm, 2 degree, the average computation time for each automatic process is listed in Table 5.5. The results illustrate that by applying the coarse-to-fine approach the computational time of constraint satisfaction has been sharply reduced. Although this coarse-to-fine strategy is not applied to the optimal solution searching phase, it reduces the computational time as well, which is a result of the fact that the number of possible solutions for each key segment pair is reduced, as illustrated in Table 5.3.

Note that the computations for each segment can be conducted parallelly inside Step II, III and V respectively, while the current implementation processes them one-by-one.

5.3 Summary

In this chapter, we proposed our computational multi-party interaction editing framework using the AHV-based representation. Its process consists of segmentation, intra-key segment editing and inter-key segment optimization. First the original action sequences are divided into key segment and transitional segment pairs. Then we propose three kinds of spatiotemporal constraints based on the multi-party interaction dictionary to compute the solution group for each key segment. Next we apply a global optimization to generate the natural looking synthesized multi-party interaction 3D video scenes.

Experiment of the proposed method has been conducted on two sets of real separately captured multi-party interaction 3D video data. Qualitative and quantitative evaluations have proved that the proposed AHV-based multi-party interaction editing algorithm can successfully synthesize smooth and natural looking multi-party interaction 3D video scenes from separately captured data, while well preserving the action dynamics of the original data.

Chapter 6

Conclusion

6.1 Summary

In this thesis we presented a novel action editing framework that synthesizes spatiotemporally synchronized multi-party interaction 3D video scenes from separately captured data. To realize that we propose the idea of Action History Volume and model the synchronization among multiple objects' body actions and gaze actions into spatiotemporal constraints that describe the multi-party interaction event. A three-step processing scheme, which includes (1) data segmentation, (2) intra-key segment editing and (3) inter-key segment optimization, has been developed to perform effective multi-party interaction synthesis work.

In details, we first introduce the Action History Volume that encodes both the spatial and temporal information to represent the spatiotemporal structure of objects' time-varying actions. By assigning labels onto the AHV surfaces and defining all types of synchronization between multiple AHVs, we build up a multi-party interaction dictionary that enables to model the synchronization of individual actions into spatiotemporal constraints.

Then to acquire the gaze vector of the object for modeling gaze actions, we propose a novel 3D non-constrained and non-contact gaze estimation method that makes full use of the multi-view video data. The algorithm for the 3D gaze estimation consists of the 3D face area detection, the symmetry plane estimation, the accurate face surface reconstruction with the symmetry prior, the super-resolution frontal face image generation and the 3D gaze estimation based on the eyeball model. Our algorithm worked stably to generate higher resolution frontal face images, and the accuracy of the last process to estimate the iris position and gaze direction was also improved. By comparing with a commercial gaze tracking device developed with conventional techniques, we have shown that our method

exceeded others in the capability of performing robust gaze estimation on freely moving object.

Finally, we proposed an executable computational scheme for synthesizing multi-party interaction scenes from separately captured 3D video data. The proposed method has been qualitatively and quantitatively evaluated with two sets of real 3D video data. Experiment results have proved the effectiveness of the proposed method in this thesis.

6.2 Future Work

In this section, we describe the future work that improves the multi-party interaction scene editing and 3D gaze estimation performance.

6.2.1 Multi-party interaction scene synthesis

For the limitations of the multi-party interaction editing algorithm, since our method only performs macroscopic position editing in the spatial editing, its performance will to some extent, rely on the degree of (un)synchronization of the original separately captured data.

If our spatiotemporal editing method can be combined with the skeletal model based editing, it will promisingly raise the flexibility of the intra-key segment editing process, and become more robust against the original data with large degree of unsynchronization. However, to achieve that we need to solve two technical problems.

Skeletal model estimation for 3D video

: First, for the unstructured data like 3D video, the skeletal model needs to be estimated for each single frame independently. Tung *et al.* has developed a method that can perform kinematic structure estimation on 3D video object with simple topologies [77]. However, for those object wearing traditional clothes that have complicated shapes and many non-rigid surfaces, accurately estimating the skeletal model is still a tough task remaining to be solved.

Surface dynamics modeling for 3D video

: Even if skeletal model of the 3D video object can be successfully estimated, that model does not account for the surface dynamics of the 3D video object. To perform skeletal model based editing while preserving the original natural surface dynamics, we need to find a way to model the surface dynamics of the 3D video object and learn how it cooperate with the changes of the skeletal model. Without such method, the skeletal based editing will inevitably destroy the surface dynamics of the original captured data.

On the other hand, because our method only requires a sequence of 3D meshes as input, it should work for (motion capture driven) skeletal data, which has a unified mesh model, as well. That is, we can further study how our spatiotemporal editing method could be combined with the skeletal based editing method, to perform effective editing on conventional CG data in producing computer animations or CG movies. For those CG data, the skeletal model is acquired during the designing step. As for the surface dynamic issue, one can use the physics simulation techniques (e.g. NVIDIA Apex clothing tool set) to generate natural looking surface motions (e.g. folding on clothes, hair swinging) accommodating the skeletal motions of the object.

Last but not least, as mentioned in Chapter 1, photometric editing of the synthesized multi-party interaction 3D video scenes could also improve the quality of the visualization results.

6.2.2 3D gaze estimation

For further studies of our 3D gaze estimation technique, we should improve the gaze estimation algorithm by exploiting the temporal information. In addition, by doing the color calibration of the multi-view cameras may also improve the quality of the generated frontal face image, thus enhance the performance of the gaze estimation processing.

Bibliography

[1] Saied Moezzi, Li-Cheng Tai, and Philippe Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, pages 18–26, 1997.

[2] Takeo Kanade, Peter Rander, and P. J. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia*, pages 34–47, 1997.

[3] Solutions, D.V.: http://www.4dviews.com/

[4] Scanner, K.M.D.L.: http://www.konicaminolta.com/

[5] Swissranger, M.: http://www.mesa-imaging.ch/

[6] Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision **47**, 7–42 (2002)

[7] for Xbox 360, M.K.: http://www.xbox.com/kinect

[8] Ikeuchi, K., Oishi, T., Takamatsu, J., Sagawa, R., Nakazawa, A., Kurazume, R., Nishino, K., Kamakura, M., Okamoto, Y.: The great buddha project: Digitally archiving, restoring, and analyzing cultural heritage objects. International Journal of Computer Vision **75**, 189–208 (2007)

[9] Kitagawa, M., Windsor, B.: MoCap for Artists: Workflow and Techniques for Motion Capture. Focal Press (2008)

[10] Vicon.: http://www.vicon.com/

[11] Rusu, R.B., Cousins, S.: 3D is here: Point Cloud Library (PCL). In: Proc. of International Conference on Robotics and Automation. Shanghai, China (2011)

[12] Kerlow, I.V.: The art of 3D computer animation and effects. John Wiley and Sons (2004)

[13] Becker, C., Reiser, M., Alkadhi, H.: Multislice CT. Springer (2008)

[14] iview: http://www.bbc.co.uk/rd/projects/iview

[15] project, E.V.: http://www.ri.cmu.edu/events/sb35/tksuperbowl.html

[16] Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000)

[17] Fujii, T., Kimoto, T., Tanimoto, M.: Ray space coding for 3d visual communication. In: Picture Coding Symposium, pp. 447–451 (1996)

[18] Levoy, M., Hanrahan, P.: Light field rendering. In: Proc. of ACM SIGGRAPH, pp. 31–42 (1996)

[19] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. IN PROCEEDINGS OF ACM SIGGRAPH 2000, pp369–374.

[20] Takai, T., Maki, A., Niinuma, K., Matsuyama, T.: Difference sphere: An approach to near light source estimation. Computer Vision and Image Understanding **113**(9), 966–978 (2009)

[21] A.F.Bobick, J.W.Davis: The recognition of human movement using temporal templates, IEEE Transactions on Pattern Analysis and Machine Intelligence 23(3), (2001)257-267.

[22] Carlos H. Morimoto and Marcio R. M. Mimica: Eye gaze tracking techniques for interactive applications, Computer Vision and Image Understanding(2005).vol.98, No.1, pp4-24.

[23] Chris Weiqle, David C. Banks: Analysis of eye-tracking experiments performed on a Tobii T 60. Visualization and Data Analysis 2008(2008), Vol.6809, No.1.

[24] Laurentini A: How far 3D shapes can be understood from 2D silhouettes. IEEE Transactions on Pattern Analysis and Machine Intelligence(1995), Vol.17(2), pp.188-195.

[25] Jean-Sébastien Franco, Edmond Boyer: Exact polyhedral visual hulls. In British Machine Vision Conference (2003), Vol. 1, pp. 329-338.

[26] Kiriakos N. Kutulakos and Steven M. Seitz: A theory of shape by space carving. In Proceedings of ICCV (1999), pp.307-314.

[27] G. Vogiatzis, P.H.S. Torr, S. Seitz and R. Cipolla: Reconstructing Relief Surfaces. In Proceedings 15th British Machine Vision Conference (2004), pp.117-126.

[28] B.Goldluecke, D.Cremers: Superresolution Texture Maps for Multiview Reconstruction. IEEE International Conference on Computer Vision(2009),pp.1-8

[29] Tony Tung, Shohei Nobuhara and Takashi Matsuyama: Simultaneous super-resolution and 3D video using graph-cuts. 26th IEEE Conference on Computer Vision and Pattern Recognition (2008), pp.1-8.

[30] H. Yamazoe, A. Utsumi, T. Yonezawa, and S. Abe: Remote gaze estimation with a single camera based on facial-feature tracking without special calibration actions. Proceedings of the 2008 symposium on Eye tracking research and applications(2008).

[31] Cootes, T.F. and Edwards, G.J. and Taylor, C.J: Active appearance models. IEEE Transactions on Pattern Analysis and Machine Intelligence 23(6), 681-685(2001).

[32] T.Ishikawa, S.Baker, I.Matthews, and T.Kanade: Passive driver gaze tracking with active appearance models. Proceedings of the 11th World Congress on Intelligent Transportation Systems(2004).

[33] E. D. Guestrin and M. Eizenman+ General theory of remote gaze estimation using the pupil center and corneal reflections. IEEE Transcations on Biomedical Engineering. 2006 Aug;53(8):1728.

[34] Y. Matsumoto and A. Zelinsky: An algorithm for realtime stereo vision implementation of head pose and gaze direction measurement. Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition(2000).

[35] Jixu Chen, Qiang Ji: Probabilistic gaze estimation without active personal calibration. CVPR 2011: 609-616

[36] Paul A Viola, Michael J. Jones: Robust Real-Time Face Detection. ICCV(2004), Vol.57 No.2, pp.137-154.

[37] Shohei Nobuhara, Yuta Kimura and Takashi Matsuyama: Object-Oriented Color Calibration of Multi-viewpoint Cameras in Sparse and Convergent Arrangement. IPSJ Transactions on Computer Vision and Applications(2010), Vol. 2, pp.132-144.

[38] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Commun. ACM 24, 381ÍC395 (1981)

[39] Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 1ÍC8 (2007)

[40] Vogiatzis, G., Torr, P.H.S., Cipolla, R.: Multi-view stereo via volumetric graph-cuts. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, pp. 391ÍC398 (2005)

[41] Shohei Nobuhara, Takashi Matsuyama: Deformable Mesh Model for Complex Multi-Object 3D Motion Estimation from Multi-Viewpoint Video, 3rd International Symposium on 3D Data Processing. Visualiztion and Transmission(2006).pp.264-271.

[42] Felzenszwalb, P., Huttenlocher, D.: Efficient belief propagation for early vision. International Journal of Computer Vision 70, 41ÍC54 (2006)

[43] Tsuyoshi Kawaguchi, Mohamed Rizon and Daisuke Hidaka: Detection of eyes from human faces by Hough transform and separability filter. Electronics and Communications in Japan(2005), Vol.88(5),pp.29-39.

[44] J. Starck, A. Hilton. Surface capture for performancebased animation. IEEE Computer Graphics and Applications, 27(3):21ÍC31, 2007.

[45] D. Vlasic, I. Baran, W. Matusik, and J. Popovic. Articulated mesh animation from multi-view silhouettes. ACM Trans. Graph., 27(3):1ÍC9, 2008

[46] T. Matsuyama, X.J. Wu, T. Takai, and T. Wada. Real-Time Dynamic 3D Object Shape Reconstruction and High-FidelityTexture Mapping for 3D Video, IEEE Trans. on Circuits and Systems for Video Technology, Vol.CSVT-14, No.3, pp.357-369, 2004.3.

[47] T. Matsuyama, X.J. Wu, T. Takai, and S. Nobuhara. Real-Time 3D Shape Reconstruction, Dynamic 3D Mesh Deformation, and High Fidelity Visualization for 3D Video. International Journal on Computer Vision and Image Understanding,Vol.96, No.3,pp.393-434, 2004.12.

[48] T. Matsuyama, S. Nobuhara, T. Takai, and T. Tung. 3D Video and Its Applications, Springer, 2012.6.

[49] T. Yamaguchi, H. Yoshimoto, S. Nobuhara, and T. Matsuyama. Cell-based 3D Video Capture of a Freely-moving Object Using Multi-viewpoint Active Cameras,IPSJ Transactions on Computer Vision and Applications, Vol.2 (2010),pp.169-184, 2010.11.10

[50] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video.

[51] J. Starck and A. Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. IEEE International Conference on Computer Vision (ICCV), pages 1387ÍC1394, 2005.

[52] J. Starck and A. Hilton. Correspondence labelling for widetimeframe free-form surface matching. Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on, pages 1ÍC8, Oct. 2007.

[53] K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In Proceedings of the Tenth European Conference on Computer Vision, volume Part II of LNCS, pages 30ÍC43, Marseille, France, October 2008. Springer-Verlag.

[54] C. Cagniart, E. Boyer, and S. Ilic. Free-Form Mesh Tracking: a Patch-Based Approach. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10), Jun 2010, San Francisco, United States.

[55] P. Huang, A. Hilton, and J. Starck. Human motion synthesis from 3d video. In CVPR, 2009.

[56] J. Starck, G. Miller, and A. Hilton, Video-Based Character Animation, Proc. ACM Symp. Computer Animation, 2005.

[57] D. Casas, M. Tejera, J. Guillemaut, A. Hilton. "Interactive Animation of 4D Performance Capture," Visualization and Computer Graphics, IEEE Transactions on , vol.19, no.5, pp.762,773, May 2013

[58] Q. Shi, S. Nobuhara, and T. Matsuyama. Augmented Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes, 3DV 2013, Washington, Seattle, USA, 2013.06.30.

[59] A. Azarbayejani, C. Wren, and A. Pentland. Real-Time 3-D Tracking of the Human Body. IN PROCEEDINGS OF IMAGE'COM, 1996.

[60] T. B. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. Computer Vision and Image Understanding. vol. 81, pp. 231-268. 2001.

[61] M. Gleicher. Motion editing with spacetime constraints. In SI3D 97: Proceedings of the 1997 symposium on Interactive 3D graphics, New York, NY, USA, 1997. ACM.

[62] J. Lee, S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. Proceedings of SIGGRAPH 99 (1999), pp. 39C48. 2.

[63] A. Brundelin, L. Williams. Motion signal processing. Proceedings of ACM SIGGRAPH pp. 97104 (1995)

[64] D. Wiley, J. Hahn. Interpolation synthesis for articulated gure motion. In: IEEE Virtual Reality International Symposium. pp. 157|160 (1997)

[65] C. Rose, M. Cohen, B. Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. IEEE Computer Graphics and Applications 18(5), 3240 (1998)

[66] L. Kovar, M. Gleicher. Automated extraction and parameterization of motions in large date sets. In: Proceedings of ACM SIGGRAPH. pp. 23(3):559|568 (2004)

[67] T. Mukai, S. Kuriyama. Geostatistical motion interpolation. ACM Transactions on Graphics 24, 3 (2005), 1062C1070. 2.

[68] R. Heck, M. Gleicher. Parametric Motion Graphs. In ACM Symposium on Interactive 3D Graphics. 2007.

[69] V.B. Zordan. Dynamic response for motion capture animation. ACM Trans. Graph., 24, 3, pp. 697-701(2005).

[70] Y. Ye, and C.K. Liu. Synthesis of responsive motion using a dynamic model. Comput. Graph. Forum, 29, 2, pp. 555-562 (2010).

[71] C. Bregler, M. Covell, and M. Slaney. Video Rewrite: Driving Visual Speech with Audio, Proc. ACM SIGGRAPH 97, pp. 1-8, 1997.

[72] A. Schodl, D. Szeliski, R. amd Salesin, and I. Essa. Video Textures, Proc. ACM SIGGRAPH 00, 2000.

[73] M. Flagg, A. Nakazawa, Q. Zhang, S.-B. Kang, Y. Ryu, I. Essa, and J. Rehg, Human Video Textures, Proc. ACM Symp. Interactive 3D Graphics, 2009.

[74] L. Kovar, M. Gleicher, and F. Pighin, Motion Graphs, Proc. ACM SIGGRAPH 02, pp. 473-482, 2002.

[75] O. Arikan and D. Forsyth. Synthesizing Constrained Motions from Examples, Proc. ACM SIGGRAPH 02, 2002.

[76] F. Xu, Y. Liu, C. Stoll, J. Tompkin, G. Bharaj, Q. Dai, H.P. Seidel, J. Kautz, and C. Theobalt, Video-Based Characters Creating New Human Performances from a Multi-View Video Database, Proc. ACM SIGGRAPH 11, 2011.

[77] T. Tung, T. Matsuyama. Topology Dictionary for 3D Video Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), Vol.34, No. 8., pp. 1645-1657, 2012.8.(Selected as Spotlight Paper)

[78] M. Flagg, A. Nakazawa, Q. Zhang, S.-B. Kang, Y. Ryu, I. Essa, and J. Rehg. Human Video Textures, Proc. ACM Symp. Interactive 3D Graphics, 2009.

[79] D. Weinland, R. Ronfard, and E. Boyer. Free view-point action recognition using motion history volumes. COMPUTER VISION AND IMAGE UNDERSTANDING (2006).

[80] Q. Shi, S. Nobuhara, and T. Matsuyama. 3D Face Reconstruction and Gaze Estimation from Multi-view Video using Symmetry Prior. IPSJ Transactions on Computer Vision and Applications, Vol.4, pp.149-160, 2012.10.1.

[81] S. Nobuhara, and T. Matsuyama. Dynamic 3D Shape from Multi-viewpoint Images using Deformale Mesh Models. Proc of 3rd International Symposium on Image and Signal Processing and Analysis, Rome, Italy. September 18-20, 2003. pp 192-197

List of Publications

Journal Paper

- Qun Shi, Shohei Nobuhara, and Takashi Matsuyama. 3D Face Reconstruction and Gaze Estimation from Multi-view Video using Symmetry Prior. *IPSJ Transactions on Computer Vision and Applications*, Vol.4 (2012), pp.149-160, 2012.10.19.

- Qun Shi, Shohei Nobuhara, and Takashi Matsuyama. Augmented Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, accepted on 2014.4.17, publication date unfixed.

International Conference

- Qun Shi, Shohei Nobuhara and Takashi Matsuyama. Augmented Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes. In *3DV 2013*, pages 1-8, Washington, Seattle, USA, 2013.06.30.

Presentation

- Qun Shi, Shohei Nobuhara, and Takashi Matsuyama. Motion History Volume for Spatiotemporal Editing of 3D Video in Multi-party Interaction Scenes. In *IPSJ SIG-CVIM: Computer Vision and Image Media*, pages 1-7, May 2013.